CPS 102

DISCRETE MATHEMATICS FOR COMPUTER SCIENCE

Spring 2009

Co-instructors: **Herbert Edelsbrunner** and **Brittany Fasy**

Table of Contents

Introduction

Meetings. We meet twice a week for lectures, on Monday and on Wednesday, from 2:50 to 4:05pm, in room D243 LSRC. We also have a recitation each week on Friday, same time and room as the lectures.

Communication. The course material will be delivered in the two weekly lectures. A written record of the lectures will be available on the web, usually a day after the lecture. The web also contains other information, such as homework assignments, solutions, useful links, etc. The main supporting text is

BOGART, STEIN, DRYSDALE. *Discrete Mathematics for Computer Science.* Key College Publishing, Emeryville, California, 2006.

Examinations. There will be a final exam (covering the material of the entire semester) and two midterm. The weighting of participation, exams, and homework used to determine your grades is

class participation	10%,
homework	30%,
midterms	30%,
final	30%.

Homework. We have six homeworks scheduled throughout this semester, one per main topic covered in the course. The solutions to each homework are due one and a half weeks after the assignment. More precisely, they are due at the beginning of the third lecture after the assignment. The sixth homework may help you prepare for the final exam and solutions will not be collected.

RULE 1. The solution to any one homework question must fit on a single page (together with the statement of the problem).

RULE 2. The discussion of questions and solutions before the due date is not discouraged, but you must formulate your own solution.

RULE 3. The deadline for turning in solutions is 10 minutes after the beginning of the lecture on the due date.

Overview. Discrete mathematics provides concepts that are fundamental to computer science but also other disciplines. This course emphasizes the computer science connection through the selection and motivation of topics, which are grouped in six major themes:

I Counting;
II Number Theory;
III Logic;
IV Induction;
V Probability;
VI Graphs.

I COUNTING

Counting things is a central problem in Discrete Mathematics. Once we can count, we can determine the likelihood of a particular even and we can estimate how long a computer algorithm takes to complete a task.

1 Sets and Lists

Sets and lists are fundamental concepts that arise in various contexts, including computer algorithms. We study basic counting problems in terms of these concepts.

Sorting. A common computational task is to rearrange elements in order. Given a linear array $A[1..n]$ of integers, rearrange them such that $A[i] \leq A[i+1]$ for $1 \leq i < n$.

```
for i = 1 to n − 1 do
  for j = i + 1 downto 2 do
    if A[j] > A[j − 1] then
      aux = A[j]; A[j] = A[j − 1]; A[j] = aux
    endif
  endfor
endfor.
```

We wish to count the number of comparisons made in this algorithm. For example, sorting an array of five elements uses 15 comparisons. In general, we make $1 + 2 + \cdots + (n-1) = \sum_{i=1}^{n-1} i$ comparisons.

Sums. We now derive a closed form for the above sum by adding it to itself. Arranging the second sum in reverse order and adding the terms in pairs, we get

$$[1 + (n-1)] + \ldots + [(n-1) + 1] \quad = \quad n(n-1).$$

Since each number of the original sum is added twice, we divide by two to obtain

$$\sum_{i=1}^{n-1} i \quad = \quad \frac{n(n-1)}{2}.$$

As with many mathematical proofs, this is not the only way to derive this sum. We can think of the sum as two sets of stairs that stack together, as in Figure 1. At the base, we have $n-1$ gray blocks and one white block. At each level, one more block changes from gray to white, until we have one gray block and $n-1$ white blocks. Together, the stairs form a rectangle divided into $n-1$ by n squares, with exactly half the squares gray and the other half white. Thus, $\sum_{i=1}^{n} i = \frac{n(n-1)}{2}$, same as before. Notice that this sum can appear in other forms, for example,

$$\begin{aligned} \sum_{i=1}^{n-1} i &= 1 + 2 + \ldots + (n-1) \\ &= (n-1) + (n-2) + \ldots + 1 \\ &= \sum_{i=1}^{n-1} (n-i). \end{aligned}$$

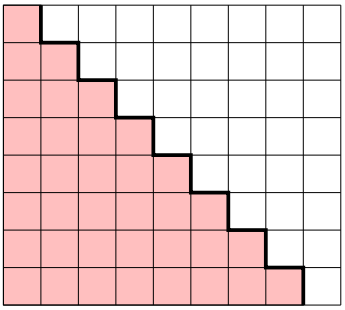


Figure 1: The number of squares in the grid is twice the sum from 1 to 8.

Sets. A *set* is an unordered collection of distinct elements. The *union* of two sets is the set of elements that are in one set or the other, that is, $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$. The *intersection* of the same two sets is the set of elements that are in both, that is, $A \cap B = \{x \mid x \in A \text{ and } x \in B\}$. We say that A and B are disjoint if $A \cap B = \emptyset$. The *difference* is the set of elements that belong to the first but not to the second set, that is, $A - B = \{x \mid x \in A \text{ and } x \notin B\}$. The *symmetric difference* is the set of elements that belong to exactly one of the two sets, that is, $A \oplus B = (A-B) \cup (B-A) = (A \cup B) - (A \cap B)$. Look at Figure 2 for a visual description of the sets that

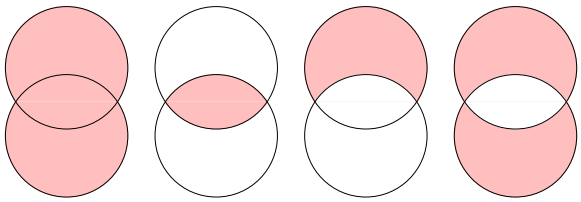


Figure 2: From left to right: the union, the intersection, the difference, and the symmetric difference of two sets represented as disks in the plane.

result from the four types of operations. The number of elements in a set A is denoted as $|A|$. It is referred to as the *size* or the *cardinality* of A. The number of elements in the union of two sets cannot be larger than the sum of the two sizes.

SUM PRINCIPLE 1. $|A \cup B| \leq |A| + |B|$ with equality if A and B are disjoint.

To generalize this observation to more than two sets, we call the sets $S_1, S_2, \ldots, S_m$ a *covering* of $S = S_1 \cup S_2 \cup \ldots \cup S_m$. If $S_i \cap S_j = \emptyset$ for all $i \neq j$, then the covering

is called a *partition*. To simplify the notation, we write $\bigcup_{i=1}^{m} S_i = S_1 \cup S_2 \cup \cdots \cup S_m$.

SUM PRINCIPLE 2. Let $S_1, S_2, \ldots, S_m$ be a covering of S. Then, $|S| \leq \sum_{i=1}^{m} |S_i|$, with equality if the covering is a partition.

Matrix multiplication. Another common computational task is the multiplication of two matrices. Assuming the first matrix is stored in a two-dimensional array $A[1..p, 1..q]$ and the second matrix is stored in $B[1..q, 1..r]$, we match up rows of A with the columns of B and form the sum of products of corresponding elements. For example, multiplying

$$A = \begin{bmatrix} 1 & 3 & 2 \\ 0 & 2 & 4 \end{bmatrix}$$

with

$$B = \begin{bmatrix} 0 & 2 & 3 \\ 1 & 2 & 5 \\ 4 & 0 & 1 \end{bmatrix}$$

results in

$$C = \begin{bmatrix} 11 & 8 & 20 \\ 18 & 4 & 14 \end{bmatrix}.$$

The algorithm we use to get C from A and B is described in the following pseudo-code.

```
for i = 1 to p do
  for j = 1 to r do
    C[i, j] = 0;
    for k = 1 to q do
      C[i, j] = C[i, j] + A[i, k] · B[k, j]
    endfor
  endfor
endfor.
```

We are interested in counting how many multiplications the algorithm takes. In the example, each entry of the result uses three multiplications. Since there are six entries in C, there are a total of $6 \cdot 3 = 18$ multiplications. In general, there are q multiplications for each of pr entries of the result. Thus, there are pqr multiplications in total. We state this observation in terms of sets.

PRODUCT PRINCIPLE 1. Let $S = \bigcup_{i=1}^{m} S_i$. If the sets $S_1, S_2, \ldots, S_m$ form a partition and $|S_i| = n$ for each $1 \leq i \leq m$ then $|S| = nm$.

We can also encode each multiplication by a triplet of integers, the row number in A, the column number in A which is also the row number in B, and the column number in B. There are p possibilities for the first number, q for the second, and r for the third number. We generalize this method as follows.

PRODUCT PRINCIPLE 2. If S is a set of lists of length m with i_j possibilities for position j, for $1 \leq j \leq m$, then $|S| = i_1 \cdot i_2 \cdot \ldots \cdot i_m = \prod_{j=1}^{m} i_j$.

We can use this rule to count the number of cartoon characters that can be created from a book giving choices for head, body, and feet. If there are p choices for the head, q choices for the body, and r choices for the legs, then there are pqr different cartoon characters we can create.

Number of passwords. We apply these principles to count the passwords that satisfy some conditions. Suppose a valid password consists of eight characters, each a digit or a letter, and there must be at least two digits. To count the number of valid passwords, we first count the number of eight character passwords without the digit constraint: $(26+10)^8 = 36^8$. Now, we subtract the number of passwords that fail to meet the digit constraint, namely the passwords with one or no digit. There are 26^8 passwords without any digits. To count the passwords with exactly one digit, we note that there are 26^7 ways to choose an ordered set of 7 letters, 10 ways to choose one digit, and 8 places to put the digit in the list of letters. Therefore, there are $26^7 \cdot 10 \cdot 8$ passwords with only one digit. Thus, there are $36^8 - 26^8 - 26^7 \cdot 10 \cdot 8$ valid passwords.

Lists. A *list* is an ordered collection of elements which are not necessarily different from each other. We note two differences between lists and sets:

(1) a list is ordered, but a set is not;

(2) a list can have repeated elements, but a set can not.

Lists can be expressed in terms of another mathematical concept in which we map elements of one set to elements of another set. A *function* f from a *domain* D to a *range* R, denoted as $f : D \to R$, associates exactly one element in R to each element $x \in D$. A list of k elements is a function $\{1, 2, \ldots, k\} \to R$. For example, the function in Figure 3 corresponds to the list $a, b, c, b, z, 1, 3, 3$. We can use the Product Principle 2 to count the number of different functions from a finite domain, D, to a finite range, R.

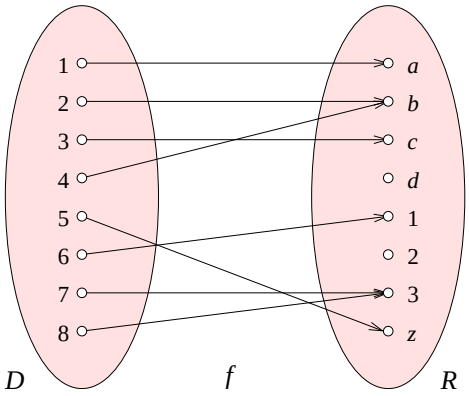


Figure 3: A function representing a list.

Specifically, we have a list of length $|D|$ with $|R|$ possibilities for each position. Hence, the number of different functions from D to R is $|R|^{|D|}$.

Bijections. The function $f : D \to R$ is *injective* or *one-to-one* if $f(x) \neq f(y)$ for all $x \neq y$. It is *surjective* or *onto* if for every $r \in R$, there exists some $x \in D$ with $f(x) = r$. The function is *bijective* or a *one-to-one correspondence* if it is both injective and surjective.

BIJECTION PRINCIPLE. Two sets D and R have the same size if and only if there exists a bijection $f : D \to R$.

Thus, asking how many bijections there are from D to R only makes sense if they have the same size. Suppose this size is finite, that is, $|D| = |R| = n$. Then being injective is the same as being bijective. To count the number of bijections, we assign elements of R to elements of D, in sequence. We have n choices for the first element in the domain, $n - 1$ choices for the second, $n - 2$ for the third, and so on. Hence the number of different bijections from D to R is $n \cdot (n - 1) \cdot \ldots \cdot 1 = n!$.

Summary. Today, we began with the building blocks of counting: sets and lists. We went through some examples using the sum and product principles: counting the number of times a loop is executed, the number of possible passwords, and the number of combinations. Finally, we talked about functions and bijections.

2 Binomial Coefficients

In this section, we focus on counting the number of ways sets and lists can be chosen from a given set.

Permutations. A *permutation* is a bijection from a finite set D to itself, $f : D \to D$. For example, the permutations of $\{1, 2, 3\}$ are: $123, 132, 213, 231, 312$, and 321. Here we list the permutations in lexicographic order, same as they would appear in a dictionary. Assuming $|D| = k$, there are $k!$ permutations or, equivalently, orderings of the set. To see this, we note that there are k choices for the first element, $k - 1$ choices for the second, $k - 2$ for the third, and so on. The total number of choices is therefore $k(k-1) \cdot \ldots \cdot 1$, which is the definition of $k!$.

Let $N = \{1, 2, \ldots, n\}$. For $k \leq n$, a *k-element permutation* is an injection $\{1, 2, \ldots, k\} \to N$. In other words, a k-element permutation is a list of k distinct elements from N. For example, the 3-element permutations of $\{1, 2, 3, 4\}$ are

123,	124,	132,	134,	142,	143,
213,	214,	231,	234,	241,	243,
312,	314,	321,	324,	341,	342,
412,	413,	421,	423,	431,	432.

There are 24 permutations in this list. There are six orderings of the subset $\{1, 2, 3\}$ in this list. In fact, each 3-element subset occurs six times. In general, we write $n^{\underline{k}}$ for the number of k-element permutations of a set of size n. We have

$$\begin{aligned} n^{\underline{k}} &= \prod_{i=0}^{k-1} (n-i) \\ &= n(n-1)\cdots(n-(k-1)) \\ &= \frac{n!}{(n-k)!}. \end{aligned}$$

Subsets. The *binomial coefficient* $\binom{n}{k}$, pronounced n choose k, is by definition the number of k-element subsets of a size n set. Since there are $k!$ ways to order a set of size k, we know that $n^{\underline{k}} = \binom{n}{k} \cdot k!$ which implies

$$\binom{n}{k} = \frac{n!}{(n-k)!k!}.$$

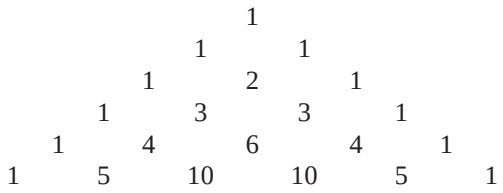
We fill out the following tables with values of $\binom{n}{k}$, where the row index is the values of n and the column index is the value of k. Values of $\binom{n}{k}$ for $k > n$ are all zero and are omitted from the table.

	0	1	2	3	4	5
0	1					
1	1	1				
2	1	2	1			
3	1	3	3	1		
4	1	4	6	4	1	
5	1	5	10	10	5	1

By studying this table, we notice several patterns.

- $\binom{n}{0} = 1$. In words, there is exactly one way to choose no item from a list of n items.
- $\binom{n}{n} = 1$. In words, there is exactly one way to choose all n items from a list of n items.
- Each row is symmetric, that is, $\binom{n}{k} = \binom{n}{n-k}$.

This table is also known as Pascal's Triangle. If we draw it symmetric between left and right then we see that each entry in the triangle is the sum of the two entries above it in the previous row.



Pascal's Relation. We express the above recipe of constructing an entry as the sum of two previous entries more formally. For convenience, we define $\binom{n}{k} = 0$ whenever $k < 0$, $n < 0$, or $n < k$.

PASCAL'S RELATION. $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$.

PROOF. We give two arguments for this identity. The first works by algebraic manipulations. We get

$$\begin{aligned} \binom{n}{k} &= \frac{(n-k)(n-1)! + k(n-1)!}{(n-k)!k!} \\ &= \frac{(n-1)!}{(n-k-1)!k!} + \frac{(n-1)!}{(n-k)!(k-1)!} \\ &= \binom{n-1}{k} + \binom{n-1}{k-1}. \end{aligned}$$

For the second argument, we partition the sets. Let $|S| = n$ and let a be an arbitrary but fixed element from S. $\binom{n}{k}$ counts the number of k-element subsets of S. To get the number of subsets that contain a, we count the $(k-1)$-element subsets of $S - \{a\}$, and to get the number of subsets that do not contain a, we count the k-element subsets

of $S - \{a\}$. The former is $\binom{n-1}{k-1}$ and the latter is $\binom{n-1}{k}$. Since the subsets that contain a are different from the subsets that do not contain a, we can use the Sum Principle 1 to get the number of k-element subsets of S equal to $\binom{n-1}{k-1} + \binom{n-1}{k}$, as required. □

Binomials. We use binomial coefficients to find a formula for $(x+y)^n$. First, let us look at an example.

$$\begin{aligned}(x+y)^2 &= (x+y)(x+y)\\ &= xx + yx + xy + yy\\ &= x^2 + 2xy + y^2.\end{aligned}$$

Notice that the coefficients in the last line are the same as in the second line of Pascal's Triangle. This is more generally the case and known as the

BINOMIAL THEOREM. $(x+y)^n = \sum_{i=0}^{n} \binom{n}{i} x^{n-i} y^i$.

PROOF. If we write each term of the result before combining like terms, we list every possible way to select one x or one y from each factor. Thus, the coefficient of $x^{n-i}y^i$ is equal to $\binom{n}{n-i} = \binom{n}{i}$. In words, it is the number of ways we can select $n-i$ factors to be x and have the remaining i factors to be y. This is equivalent to selecting i factors to be y and have the remaining factors be x. □

Corollaries. The Binomial Theorem can be used to derive a number of other interesting sums. We prove three such consequences.

COROLLARY 1. $\sum_{i=0}^{n} \binom{n}{i} = 2^n$.

PROOF. Let $x = y = 1$. Then, by the Binomial Theorem we have

$$(1+1)^n = \sum_{i=0}^{n} \binom{n}{i} 1^{n-i} 1^i.$$

This implies the claimed identity. □

COROLLARY 2. $\sum_{j=k}^{n} \binom{j}{k} = \binom{n+1}{k+1}$.

PROOF. We use Pascal's Relation to prove this identity. It is instructive to trace our steps graphically, in the triangle above. In a first step, we replace $\binom{n+1}{k+1}$ by $\binom{n}{k}$ and $\binom{n}{k+1}$. Keeping the first term, we replace the second, $\binom{n}{k+1}$, by $\binom{n-1}{k}$ and $\binom{n-1}{k+1}$. Repeating this operation, we finally replace $\binom{k+1}{k+1}$ by $\binom{k}{k} = 1$ and $\binom{k}{k+1} = 0$. In other words, $\binom{n+1}{k+1}$ is equal to the sum of the $\binom{j}{k}$ for j running from n down to k. □

COROLLARY 3. $\sum_{i=1}^{n} i^2 = \frac{n^3}{3} + \frac{n^2}{2} + \frac{n}{6}$.

PROOF. We first express the summands in terms of binomial coefficients and then use Corollary 2 to get the result.

$$\begin{aligned}\sum_{i=1}^{n} i^2 &= 2\sum_{i=1}^{n} \frac{i^2 - i}{2} + \sum_{i=1}^{n} i\\ &= 2\sum_{i=1}^{n} \binom{i}{2} + \sum_{i=1}^{n} \binom{i}{1}\\ &= 2\binom{n+1}{3} + \binom{n+1}{2}\\ &= \frac{2(n+1)n(n-1)}{1\cdot 2\cdot 3} + \frac{(n+1)n}{1\cdot 2}\\ &= \frac{n^3 - n}{3} + \frac{n^2 + n}{2}.\end{aligned}$$

This implies the claimed identity. □

Summary. The binomial coefficient, $\binom{n}{k}$, counts the different ways we can choose k elements from a set of n. We saw how it can be used to compute $(x+y)^n$. We proved several corollaries and saw that describing the identities as counting problems can lead us to different, sometimes simpler proofs.

3 Equivalence Relations

Equivalence relations are a way to partition a set into subsets of equivalent elements. Being equivalent is then interpreted as being the same, such as different views of the same object or different ordering of the same elements, etc. By counting the equivalence classes, we are able to count the items in the set that are different in an essential way.

Labeling. To begin, we ask how many ways are there to label three of five elements red and the remaining two elements blue? Without loss of generality, we can call our elements A, B, C, D, E. A labeling is an function that associates a color to each element. Suppose we look at a permutation of the five elements and agree to color the first three red and the last two blue. Then the permutation $ABDCE$ would correspond to coloring A, B, D red and C, E blue. However, we get the same labeling with other permutations, namely

$$\begin{array}{lll} ABD;CE & BAD;CE & DAB;CE \\ ABD;EC & BAD;EC & DAB;EC \\ ADB;CE & BDA;CE & DBA;CE \\ ADB;EC & BDA;EC & DBA;EC\ . \end{array}$$

Indeed, we have $3!2! = 12$ permutations that give the same labeling, simply because there are $3!$ ways to order the red elements and $2!$ ways to order the blue elements. Similarly, every other labeling corresponds to 12 permutations. In total, we have $5! = 120$ permutations of five elements. The set of 120 permutations can thus be partitioned into $\frac{120}{12} = 10$ blocks such that any two permutations in the same block give the same labeling. Any two permutations from different blocks give different labelings, which implies that the number of different labelings is 10. More generally, the number of ways we can label k of n elements red and the remaining $n-k$ elements blue is $\frac{n!}{k!(n-k)!} = \binom{n}{k}$. This is also the number of k-element subsets of a set of n elements.

Now suppose we have three labels, red, green, and blue. We count the number of different labelings by dividing the total number of orderings by the orderings within in the color classes. There are $n!$ permutations of the n elements. We want i elements red, j elements blue, and $k = n - i - j$ elements green. We agree that a permutation corresponding to the labeling we get by coloring the first i elements red, the next j elements blue, and the last k elements green. The number of repeated labelings is thus $i!$ times $j!$ times $k!$ and we have $\frac{n!}{i!j!k!}$ different labelings.

Equivalence relations. We now formalize the above method of counting. A *relation* on a set S is a collection R of ordered pairs, (x, y). We write $x \sim y$ if the pair (x, y) is in R. We say that a relation is

- *reflexive* if $x \sim x$ for all $x \in S$;
- *symmetric* if $x \sim y$ implies $y \sim x$;
- *transitive* if $x \sim y$ and $y \sim z$ imply $x \sim z$.

We say that the relation is an *equivalence relation* if R is reflexive, symmetric, and transitive. If S is a set and R an equivalence relation on S, then the *equivalence class* of an element $x \in S$ is

$$[x] \quad = \quad \{y \in S \mid y \sim x\}.$$

We note here that if $x \sim y$ then $[x] = [y]$. In the above labeling example, S is the set of permutations of the elements A, B, C, D, E and two permutations are equivalent if they give the same labeling. Recalling that we color the first three elements red and the last two blue, the equivalence classes are $[ABC; DE]$, $[ABD; CE]$, $[ABE; CD]$, $[ACD; BE]$, $[ACE; BD]$, $[ADE; BC]$, $[BCD; AE]$, $[BCE; AD]$, $[BDE; AC]$, $[CDE; AB]$.

Not all relations are equivalence relations. Indeed, there are relations that have none of the above three properties. There are also relations that satisfy any subset of the three properties but none of the rest.

An example: modular arithmetic. We say an integer a is *congruent* to another integer b modulo a positive integer n, denoted as $a = b \bmod n$, if $b - a$ is an integer multiple of n. To illustrate this definition, let $n = 3$ and let S be the set of integers from 0 to 11. Then $x = y \bmod 3$ if x and y both belong to $S_0 = \{0, 3, 6, 9\}$ or both belong to $S_1 = \{1, 4, 7, 10\}$ or both belong to $S_2 = \{2, 5, 8, 11\}$. This can be easily verified by testing each pair. Congruence modulo 3 is in fact an equivalence relation on S. To see this, we show that congruence modulo 3 satisfies the three required properties.

reflexive. Since $x - x = 0 \cdot 3$, we know that $x = x \bmod 3$.

symmetric. If $x = y \bmod 3$ then x and y belong to the same subset S_i. Hence, $y = x \bmod 3$.

transitive. Let $x = y \bmod 3$ and $y = z \bmod 3$. Hence x and y belong to the same subset S_i and so do y and z. It follows that x and z belong to the same subset.

More generally, congruence modulo n is an equivalence relation on the integers.

Block decomposition. An equivalence class of elements is sometimes called a *block*. The importance of equivalence relations is based on the fact that the blocks partition the set.

THEOREM. Let R be an equivalence relation on some set S. Then the blocks $S_x = \{y \in S \mid x \sim y, y \in S\}$ for all $x \in S$ partition S.

PROOF. In order to prove that $\bigcup_x S_x = S$, we need to show two things, namely $\bigcup_{x \in S} S_x \subseteq S$ and $\bigcup_{x \in S} S_x \supseteq S$. Each S_x is a subset of S which implies the first inclusion. Furthermore, each $x \in S$ belongs to S_x which implies the second inclusion. Additionally, if $S_x \neq S_y$, then $S_x \cap S_y = \emptyset$ since $z \in S_x$ implies $z \sim x$, which means that $S_x = S_z$, which means that $S_z \neq S_y$. Therefore, z is not related to y, and so $z \notin S_y$. □

Symmetrically, a partition of S defines an equivalence relation. If the blocks are all of the same size then it is easy to count them.

QUOTIENT PRINCIPLE. If a set S of size p can be partitioned into q classes of size r each, then $p = qr$ or, equivalently, $q = \frac{p}{r}$.

Multisets. The difference between a set and a *multiset* is that the latter may contain the same element multiple times. In other words, a multiset is an unordered collection of elements, possibly with repetitions. We can list the repetitions,

$$\langle\langle c, o, l, o, r\rangle\rangle$$

or we can specify the multiplicities,

$$m(c) = 1, m(o) = 2, m(r) = 1.$$

The *size* of a multiset is the sum of the multiplicities. We show how to count multisets by considering an example, the ways to distribute k (identical) books among n (different) shelves. The number of ways is equal to

- the number of size-k multisets of the n shelves;
- the number of ways to write k as a sum of n non-negative integers.

We count the ways to write k as a sum of n non-negative integers as follows. Choose the first integer of the sum to be p. Now we have reduced the problem to counting the ways to write $k - p$ as the sum of $n - 1$ non-negative integers. For small values of n, we can do this.

For example, let $n = 3$. Then, we have $p + q + r = k$. The choices for p are from 0 to k. Once p is chosen, the choices for q are fewer, namely from 0 to $k - p$. Finally, if p and q are chosen then r is determined, namely $r = k - p - q$. The number of ways to write k as a sum of three non-negative integers is therefore

$$\begin{aligned}\sum_{p=0}^{k}\sum_{q=0}^{k-i} 1 &= \sum_{p=0}^{k}(k-p+1)\\ &= \sum_{p=1}^{k+1} p\\ &= \binom{k+2}{2}.\end{aligned}$$

There is another (simpler) way of finding this solution. Suppose we line up our n books, then place $k - 1$ dividers between them. The number of books between the i-th and the $(i - 1)$-st dividers is equal to the number of books on the i-th shelf; see Figure 4. We thus have $n + k - 1$ objects, k books plus $n - 1$ dividers. The number of ways to

Figure 4: The above arrangement of books and blocks represents two books placed on the first and last shelves, and one book on the second shelf. As a sum, this figure represents $2 + 1 + 0 + 2$.

choose $n - 1$ dividers from $n + k - 1$ objects is $\binom{n+k-1}{n-1}$. We can easily see that this formula agrees with the result we found for $n = 3$.

Summary. We defined relations and equivalence relations, investigating several examples of both. In particular, modular arithmetic creates equivalence classes of the integers. Finally, we looked at multisets, and saw that counting the number of size-k multisets of n elements is equal to the number of ways to write k as a sum of n non-negative integers.

First Homework Assignment

Write the solution to each question on a single page. The deadline for handing in solutions is January 26.

Question 1. ($20 = 10 + 10$ points). If n basketball teams play each other team exactly once, how many games will be played in total? If the teams then compete in a single elimination tournament (similar to March Madness), how many additional games are played?

Question 2. ($20 = 10 + 10$ points).

(a) (Problem 1.2-7 in our textbook). Let $|D| = |R| = n$. Show that the following statement is true: The function $f : D \to R$ is surjective if and only if f is injective.

(b) Is the function $f : \mathbb{R} \to \mathbb{R}$ defined by $f(x) = 3x + 2$ a bijection? Prove or give a counterexample.

Question 3. ($20 = 6 + 7 + 7$ points).

(a) What is the coefficient of the x^8 term of $(x - 2)^{30}$?

(b) What is the coefficient of the $x^i y^j z^k$ term of $(x + y + z)^n$?

(c) Show that $\binom{n}{k} = \binom{n}{n-k}$.

Question 4. ($20 = 6+7+7$ points). For (a) and (b), prove or disprove that the relations given are equivalence relations. For (c), be sure to justify your answer.

(a) Choose some $k \in \mathbb{Z}$. Let $x, y \in \mathbb{Z}$. We say $x \sim y$ if $x \equiv y \mod k$.

(b) Let x, y be positive integers. We say $x \sim y$ if the greatest common factor of x and y is greater than 1.

(c) How many ways can you distribute k identical cookies to n children?

II NUMBER THEORY

We use the need to send secret messages as the motivation to study questions in number theory. The main tool for this purpose is modular integer arithmetic.

4 Modular Arithmetic

We begin the chapter on number theory by introducing modular integer arithmetic. One of its uses is in the encryption of secret messages. In this section, all numbers are integers.

Private key cryptography. The problem of sending secret messages is perhaps as old as humanity or older. We have a *sender* who attempts to encrypt a message in such a way that the intended *receiver* is able to decipher it but any possible *adversary* is not. Following the traditional protocol, the sender and receiver agree on a secret code ahead of time, and they use it to both encrypt and decipher the message. The weakness of the method is the secret code, which may be stolen or cracked.

As an example, consider *Ceasar's cipher*, which consists of shifting the alphabet by some fixed number of positions, e.g.,

$$\begin{array}{ccccccccc} A & B & C & \dots & V & W & X & Y & Z \\ \downarrow & \downarrow & \downarrow & \dots & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ E & F & G & \dots & Z & A & B & C & D. \end{array}$$

If we encode the letters as integers, this is the same as adding a fixed integer but then subtracting 26, the number of letters, if the sum exceeds this number. We consider this kind of integer arithmetic more generally.

Public key cryptography. Today, we use more powerful encryption methods that give a more flexible way to transmit secret information. We call this *public key cryptography* which roughly works as follows. As before, we have a sender, called Alice, and a receiver, called Bob. Both Alice and Bob have a *public key*, KP_A and KP_B, which they publish for everyone to see, and a *secret key*, KS_A and KS_B, which is only known to themselves. They do not exchange the secret key even among each other. The keys are used to change messages so we can think of them as functions. The function that corresponds to the public and the secret keys are inverses of each other, that is,

$$\begin{aligned} S_A(P_A(x)) &= P_A(S_A(x)) = x; \\ S_B(P_B(x)) &= P_B(S_B(x)) = x. \end{aligned}$$

The crucial point is that P_A is easy to compute for everybody and S_A is easy to compute for Alice but difficult for everybody else, including Bob. Symmetrically, P_B is easy for everybody but S_B is easy only for Bob. Perhaps this sound contradictory since everybody knows P_A and S_A is just its inverse, but it turns out that there are pairs of functions that satisfy this requirement. Now, if Alice wants to send a message to Bob, she proceeds as follows:

1. Alice gets Bob's public key, P_B.
2. Alice applies it to encrypt her message, $y = P_B(x)$.
3. Alice sends y to Bob, publically.
4. Bob applies $S_B(y) = S_B(P_B(x)) = x$.

We note that Alice does not need to know Bob's secret key to encrypt her message and she does not need secret channels to transmit her encrypted message.

Arithmetic modulo n. We begin by defining what it means to take one integer, m, modulo another integer, n.

DEFINITION. Letting $n \leq 1$, $m \bmod n$ is the smallest integer $r \geq 0$ such that $m = nq + r$ for some integer q.

Given m and $n \geq 1$, it is not difficult to see that q and r exist. Indeed, n partitions the integers into intervals of length n:

$$\dots, -n, \dots, 0, \dots, n, \dots, 2n, \dots$$

The number m lies in exactly one of these intervals. More precisely, there is an integer q such that $qn \leq m < ((q + 1)n$. The integer r is the amount by which m exceeds qn, that is, $r = m - qn$. We see that q and r are unique, which is known as

EUCLID'S DIVISION THEOREM. Letting $n \geq 1$, for every m there are unique integers q and $0 \leq r < n$ such that $m = nq + r$.

Computations. It is useful to know that modulos can be taken anywhere in the calculation if it involves only addition and multiplication. We state this more formally.

LEMMA 1. Letting $n \geq 1$, $i \bmod n = (i + kn) \bmod n$.

This should be obvious because adding k times n moves the integer i to the right by k intervals but maintains its relative position within the interval.

LEMMA 2. Letting $n \geq 1$, we have

$$\begin{aligned} (i + j) \bmod n &= (i \bmod n) + (j \bmod n) \bmod n; \\ (i \cdot j) \bmod n &= (i \bmod n) \cdot (j \bmod n) \bmod n. \end{aligned}$$

PROOF. By Euclid's Division Theorem, there are unique integers q_i, q_j and $0 \le r_i, r_j < n$ such that

$$\begin{aligned} i &= q_i n + r_i; \\ j &= q_j n + r_j. \end{aligned}$$

Plugging this into the left hand side of the first equation, we get

$$\begin{aligned} (i+j) \bmod n &= (q_i + q_j)n + (r_i + r_j) \bmod n \\ &= (r_i + r_j) \bmod n \\ &= (i \bmod n) + (j \bmod n) \bmod n. \end{aligned}$$

Similarly, it is easy to show that $(ij) \bmod n = (r_i r_j) \bmod n$, which implies the second equation. □

Algebraic structures. Before we continue, we introduce some notation. Let $\mathbb{Z}_n = \{0, 1, \ldots, n-1\}$ and write $+_n$ for addition modulo n. More formally, we have an operation that maps two numbers, $i \in \mathbb{Z}_n$ and $j \in \mathbb{Z}_n$, to their sum, $i +_n j = (i+j) \bmod n$. This operation satisfies the following four properties:

- it is *associative*, that is, $(i +_n j) +_n k = i +_n (j +_n k)$ for all $i, j, k \in \mathbb{Z}_n$;
- $0 \in \mathbb{Z}_n$ is the *neutral element*, that is, $0 +_n i = i$ for all $i \in \mathbb{Z}_n$;
- every $i \in \mathbb{Z}_n$ has an *inverse element* i', that is, $i +_n i' = 0$;
- it is *commutative*, that is, $i +_n j = j +_n i$ for all $i, j \in \mathbb{Z}_n$.

The first three are the defining property of a *group*, and if the fourth property is also satisfied we have a *commutative* or *Abelian group*. Thus, $(\mathbb{Z}_n, +_n)$ is an Abelian group. We have another operation mapping i and j to their product, $i \cdot_n j = (ij) \bmod n$. This operation has a similar list of properties:

- it is *associative*, that is, $(i \cdot_n j) \cdot_n k = i \cdot_n (j \cdot_n k)$ for all $i, j, k \in \mathbb{Z}_n$;
- $1 \in \mathbb{Z}_n$ is the *neutral element*, that is, $1 \cdot_n i = i$ for all $i \in \mathbb{Z}_n$;
- it is *commutative*, that is, $i \cdot_n j = j \cdot_n i$ for all $i, j \in \mathbb{Z}_n$.

Under some circumstances, we also have inverse elements but not in general. Hence, $(\mathbb{Z}_n, \cdot_n)$ is generally not a group. Considering the interaction of the two operations, we note that

- multiplication *distributes* over addition, that is, $i \cdot_n (j +_n k) = (i \cdot_n j) +_n (i \cdot_n k)$ for all $i, j, k \in \mathbb{Z}_n$.

These are the eight defining properties of a *commutative ring*. Had we also a multiplicative inverse for every non-zero element then the structure would be called a *field*. Hence, $(\mathbb{Z}_n, +_n, \cdot_n)$ is a commutative ring. We will see in the next section that it is a field if n is a prime number.

Addition and multiplication modulo n. We may be tempted to use modular arithmetic for the purpose of transmitting secret messages. As a first step, the message is interpreted as an integer, possibly a very long integer. For example, we may write each letter in ASCII and read the bit pattern as a number. Then we concatenate the numbers. Now suppose Alice and Bob agree on two integers, $n \ge 1$ and a, and they exchange messages using

$$\begin{aligned} P(x) &= x +_n a; \\ S(y) &= y +_n (-a) = y -_n a. \end{aligned}$$

This works fine but not as a public key cryptography system. Knowing that P is the same as adding a modulo n, it is easy to determine its inverse, S. Alternatively, let us use multiplication instead of addition,

$$\begin{aligned} P(x) &= x \cdot_n a; \\ S(y) &= y \cdot_n (-a) = y :_n a. \end{aligned}$$

The trouble now is that division modulo n is not as straightforward an operation as for integers. Indeed, if $n = 12$ and $a = 4$, we have $0 \cdot 4 = 3 \cdot 4 = 6 \cdot 4 = 9 \cdot 4 = 0 \bmod n$. Since multiplication with 4 is not injective, the inverse operation is not well defined. Indeed, $0 :_n 4$ could be 0, 3, 6, or 9.

Summary. We learned about private and public key cryptography, ways to to send a secret message from a sender to a receiver. We also made first steps into number theory, introducing modulo arithmetic and Euclid's Division Theorem. We have seem that addition and multiplication modulo n are both commutative and associative, and that multiplication distributes over addition, same as in ordinary integer arithmetic.

5 Inverses

In this section, we study under which conditions there is a multiplicative inverse in modular arithmetic. Specifically, we consider the following four statements.

I. The integer a has a multiplicative inverse in $\mathbb{Z}_n$.

II. The linear equation $a \cdot_n x = b$ has a solution in $\mathbb{Z}_n$.

III. The linear equation $ax + ny = 1$ has a solution in the integers.

IV. The integers a and n are relative prime.

We will see that all four statements are equivalent, and we will prove all necessary implications to establish this, except for one, which we will prove in the next section.

Examples. Before starting the proofs, we compute multiplicative inverses for a few values of n and a; see Table 1. Except for $a = 0$, all values of a have multiplicative inverses if $n = 2, 3, 5, 7$ but not if $n = 4, 6, 8, 9$. In the latter case, we have multiplicative inverses for some values of a but not for all. We will later find out that the characterizing condition for the existence of the multiplicative inverse is that n and a have no non-trivial common divisor.

$n=2$	a	0	1						
	a'		1						
$n=3$	a	0	1	2					
	a'		1	2					
$n=4$	a	0	1	2	3				
	a'		1		3				
$n=5$	a	0	1	2	3	4			
	a'		1	2	3	4			
$n=6$	a	0	1	2	3	4	5		
	a'		1				5		
$n=7$	a	0	1	2	3	4	5	6	
	a'		1	4	5	2	3	6	
$n=8$	a	0	1	2	3	4	5	6	7
	a'		1		3		5		7
$n=9$	a	0	1	2	3	4	5	6	7	8
	a'		1	5		7	2		4	8

Table 1: Values of n for which a has a multiplicative inverse a'. Black entries indicate the inverse does not exist.

Linear equations modulo n. Here we prove I $\iff$ II. The *multiplicative inverse* of an integer $a \in \mathbb{Z}_n$ is another integer $a' \in \mathbb{Z}_n$ such that $a' \cdot_n a = a \cdot_n a' = 1$. We note that the multiplicative inverse is unique, if it exists. Indeed, if $a'' \cdot_n a = 1$ then we can multiply with a' from the right and get $a' \cdot_n (a \cdot_n a') = a'' \cdot_n (a \cdot_n a')$ and therefore $a' = a''$. If a has a multiplicative inverse, we can use it to solve a linear equation. Multiplying with the inverse from the left and using associativity, we get

$$\begin{aligned} a \cdot_n x &= b; \\ (a' \cdot_n a) \cdot_n x &= a' \cdot_n b; \\ x &= a' \cdot_n b. \end{aligned}$$

Since the multiplicative inverse is unique, so is the solution $x = a' \cdot_n b$ to the linear equation. We thus proved a little bit more than I $\implies$ II, namely also the uniqueness of the solution.

A. If a has a multiplicative inverse a' in $\mathbb{Z}_n$ then for every $b \in \mathbb{Z}_n$, the equation $a \cdot_n x = b$ has the unique solution $x = a' \cdot_n b$.

Every implication has an equivalent contrapositive form. For a statement I $\implies$ II this form is $\neg$II $\implies \neg$I. We state the contrapositive form in this particular instance.

A'. If $a \cdot_n x = b$ has no solution in $\mathbb{Z}_n$ then a does not have a multiplicative inverse.

To prove A' we just need to assume that it is false, that is, that $\neg$II and I both hold. But if we have I then we also have II. Now we have $\neg$II as well as II. But this is a contradiction with they cannot both be true. What we have seen here is a very simple version of a proof by contradiction. More complicated versions will follow later.

By setting $b = 1$, we get $x = a'$ as a solution to $a \cdot_n x = 1$. In other words, $a' \cdot_n a = a \cdot_n a' = 1$. Hence, II $\implies$ I. This particuar implication is called the converse of I $\implies$ II, which should not be confused with the contrapositive. The converse is a new, different statement, while the contrapositive is logically eqivalent to the original implication, no matter what the specifics of the implication are.

Linear equations in two variables. Here we prove II $\iff$ III. Recall that $a \cdot_n x = 1$ is equivalent to $ax \bmod n = 1$. Writing $ax = qn + r$ with $0 \le r < n$, we see that $ax \bmod n = 1$ is equivalent to the existence of an integer q such that $ax = qn + 1$. Writing $y = -q$ we get

$$ax + ny = 1.$$

All steps in the above derivation are reversible. Hence, we proved that II is equivalent to III. We state the specific result.

B. The equation $a \cdot_n x = b$ has a solution in $\mathbb{Z}_n$ iff there exist integers x and y such that $ax + ny = 1$.

Implications are transitive, that is, if I implies II and II implies III then I implies III. We can do the same chain of implications in the other direction as well. Hence, if I $\iff$ II and II $\iff$ III, as we have established above, we also have I $\iff$ III. We again state this specific result for clarity.

C. The integer a has a multiplicative inverse in $\mathbb{Z}_n$ iff there exist integers x and y such that $ax + ny = 1$.

Greatest common divisors. Here we prove III $\implies$ IV. We will prove IV $\implies$ III later. We say an integer i *factors* another integer j if j/i is an integer. Furthermore, j is a *prime number* if its only factors are $\pm j$ and ± 1. The *greatest common divisor* of two integers j and k, denoted as $\gcd(j, k)$, is the largest integer d that is a factor of both. We say j and k and *relative prime* if $\gcd(j, k) = 1$.

D. Given integers a and n, if there exist integers x and y such that $ax + ny = 1$ then $\gcd(a, n) = 1$.

PROOF. Suppose $\gcd(a, n) = k$. Then we can write $a = ik$ and $n = jk$. Substituting these into the linear equation gives

$$\begin{aligned} 1 &= ax + ny \\ &= k(ix + jy). \end{aligned}$$

But then k is a factor of 1 and therefore $k = \pm 1$. This implies that the only common factors of a and n are ± 1 and therefore $\gcd(a, n) = 1$. □

Summary. We have proved relationships between the statements I, II, III, IV; see Figure 5. We will see later that

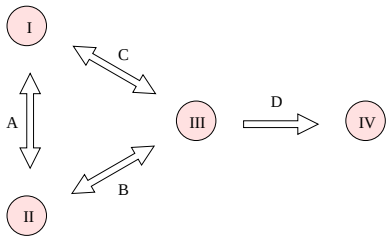


Figure 5: Equivalences between statements.

the implication proved by D can also be reversed. Thus computing the greatest common divisor gives a test for the existence of a multiplicative inverse.

6 Euclid's Algorithm

In this section, we present Euclid's algorithm for the greatest common divisor of two integers. An extended version of this algorithm will furnish the one implication that is missing in Figure 5.

Reduction. An important insight is Euclid's Division Theorem stated in Section 4. We use it to prove a relationship between the greatest common divisors of numbers j and k when we replace k by its remainder modulo j.

LEMMA. Let $j, k, q, r > 0$ with $k = jq + r$. Then $\gcd(j, k) = \gcd(r, j)$.

PROOF. We begin by showing that every common factor of j and k is also a factor of r. Letting $d = \gcd(j, k)$ and writing $j = Jd$ and $k = Kd$, we get

$$r \;=\; k - jq \;=\; (K - Jq)d.$$

We see that r can be written as a multiple of d, so d is indeed a factor of r. Next, we show that every common factor of r and j is also a factor of k. Letting $d = \gcd(r, j)$ and writing $r = Rd$ and $j = Jd$, we get

$$k \;=\; jq + r \;=\; (Jq + R)d.$$

Hence, d is indeed a factor of k. But this implies that d is a common factor of j and k iff it is a common factor of r and j. □

Euclid's gcd algorithm. We use the Lemma to compute the greatest common divisor of positive integers j and k. The algorithm is recursive and reduces the integers until the remainder vanishes. It is convenient to assume that both integers, j and k, are positive and that $j \leq k$.

```
integer GCD(j, k)
  q = k div j; r = k − jq;
  if r = 0 then return j
           else return GCD(r, j)
  endif.
```

If we call the algorithm for $j > k$ then the first recursive call is for k and j, that is, it reverses the order of the two integers and keeps them ordered as assumed from then on. Note also that $r < j$. In words, the first parameter, j, shrinks in each iterations. There are only a finite number of non-negative integers smaller than j which implies that after a finite number of iterations the algorithm halts with $r = 0$. In other words, the algorithm terminates after a finite number of steps, which is something one should always check, in particular for recursive algorithms.

Last implication. We modify the algorithm so it also returns the integers x and y for which $\gcd(j, k) = jx+ky$. This provides the missing implication in Figure 5.

D'. If $\gcd(a, n) = 1$ then the linear equation $ax+ny = 1$ has a solution.

This finally verifies that the gcd is a test for the existence of a multiplicative inverse in modular arithmetic. More specifically, $x \bmod n$ *is* the multiplicative inverse of a in $\mathbb{Z}_n$. Do you see why? We can thus update the relationship between the statements I, II, III, IV listed at the beginning of Section 5; see Figure 6.

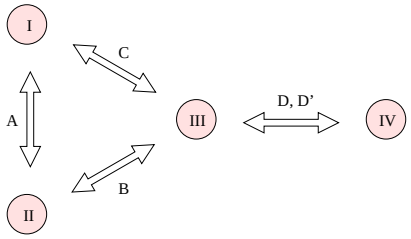


Figure 6: Equivalences between the statements listed at the beginning of Section 5.

Extended gcd algorithm. If $r = 0$ then the above algorithm returns j as the gcd. In the extended algorithm, we also return $x = 1$ and $y = 0$. Now suppose $r > 0$. In this case, we recurse and get

$$\begin{aligned} \gcd(r, j) &= rx' + jy' \\ &= (k - jq)x' + jy' \\ &= j(y' - qx') + kx'. \end{aligned}$$

We thus return $g = \gcd(r, j)$ as well as $x = y' - qx'$ and $y = x'$. As before, we assume $0 < j \leq k$ when we call the algorithm.

```
integer³ xGCD(j, k)
  q = k div j; r = k − jq;
  if r = 0 then return (j, 1, 0)
           else (g, x', y') = xGCD(r, j);
                return (g, y' − qx', x')
  endif.
```

To illustrate the algorithm, we run it for $j = 14$ and $k = 24$. The values of $j, k, q, r, g = \gcd(j,k), x, y$ at the various levels of recursion are given in Table 2.

j	k	q	r	g	x	y
14	24	1	10	2	-5	3
10	14	1	4	2	3	-2
4	10	2	2	2	-2	1
2	4	2	0	2	1	0

Table 2: Running the extended gcd algorithm on $j = 14$ and $k = 24$.

Computing inverses. We have established that the integer a has a multiplicative inverse in $\mathbb{Z}_n$ iff $\gcd(a,n) = 1$. Assuming $n = p$ is a prime number, this is the case whenever $a < p$ is positive.

COROLLARY. If p is prime then every non-zero $a \in \mathbb{Z}_p$ has a multiplicative inverse.

It is straightforward to compute the multiplicative inverse using the extended gcd algorithm. As before, we assume p is a prime number and $0 < a < p$.

```
integer INVERSE(a, p)
  (g, x, y) = xGCD(a, p);
  assert g = 1; return x mod p.
```

The assert statement makes sure that a and p are indeed relative prime, for else the multiplicative inverse would not exist. We have seen that x can be negative so it is necessary to take x modulo p before we report it as the multiplicative inverse.

Multiple moduli. Sometimes, we deal with large integers, larger then the ones that fit into a single computer word (usually 32 or 64 bits). In this situation, we have to find a representation that spreads the integer over several words. For example, we may represent an integer x by its remainders modulo 3 and modulo 5, as shown in Table 3. We see that the first 15 non-negative integers correspond to different pairs of remainders. The generalization of this insight to relative prime numbers m and n is known as the

x	0	1	2	3	4	...	13	14	15
$x \bmod 3$	0	1	2	0	1	...	1	2	0
$x \bmod 5$	0	1	2	3	4	...	3	4	0

Table 3: Mapping the integers from 0 to 15 to pairs of remainders after dividing with 3 and with 5.

CHINESE REMAINDER THEOREM. Let $m, n > 0$ be relative prime. Then for every $a \in \mathbb{Z}_m$ and $b \in \mathbb{Z}_n$, the system of two linear equations

$$\begin{aligned} x \bmod m &= a; \\ x \bmod n &= b \end{aligned}$$

has a unique solution in $\mathbb{Z}_{mn}$.

There is a further generalization to more then two moduli that are pairwise relative prime. The proof of this theorem works as suggested by the example, namely by showing that $f : \mathbb{Z}_{mn} \to \mathbb{Z}_m \times \mathbb{Z}_n$ defined by

$$f(x) = (x \bmod m, x \bmod n)$$

is injective. Since both $\mathbb{Z}_{mn}$ and $\mathbb{Z}_m \times \mathbb{Z}_n$ have size mn, this implies that f is a bijection. Hence, $(a,b) \in \mathbb{Z}_m \times \mathbb{Z}_n$ has a unique preimage, the solution of the two equations.

To use this result, we would take two large integers, x and y, and represent them as pairs, $(x \bmod m, x \bmod n)$ and $(x \bmod m, x \bmod n)$. Arithmetic operations can then be done on the remainders. For example, x times y would be represented by the pair

$$\begin{aligned} xy \bmod m &= [(x \bmod m)(y \bmod m)] \bmod m; \\ xy \bmod n &= [(x \bmod n)(y \bmod n)] \bmod n. \end{aligned}$$

We would choose m and n small enough so that multiplying two remainders can be done using conventional, single-word integer multiplication.

Summary. We discussed Euclid's algorithm for computing the greatest common divisor of two integers, and its extended version which provides the missing implication in Figure 5. We have also learned the Chinese Remainder Theorem which can be used to decompose large integers into digestible junks.

7 RSA Cryptosystem

Addition and multiplication modulo n do not offer the computational difficulties needed to build a viable cryptographic system. We will see that exponentiation modulo n does.

Operations as functions. Recall that $+_n$ and $\cdot_n$ each read two integers and return a third integer. If we fix one of the two input integers, we get two functions. Specifically, fixing $a \in \mathbb{Z}_n$, we have functions $A : \mathbb{Z}_n \to \mathbb{Z}_n$ and $M : \mathbb{Z}_n \to \mathbb{Z}_n$ defined by

$$\begin{aligned} A(x) &= x +_n a; \\ M(x) &= x \cdot_n a; \end{aligned}$$

see Table 4. Clearly, A is injective for every choice of

x	0	1	2	3	4	5
$A(x)$	2	3	4	5	0	1
$M(x)$	0	2	4	0	2	4

Table 4: The function A defined by adding $a = 2$ modulo $n = 6$ is injective. In contrast, the function M defined by multiplying with $a = 2$ is not injective.

$n > 0$ and $a \in \mathbb{Z}_n$. On the other hand, M is injective iff $\gcd(a, n) = 1$. In particular, M is injective for every non-zero $a \in \mathbb{Z}_n$ if n is prime.

Exponentiation. Yet another function we may consider is taking a to the x-th power. Let $E : \mathbb{Z}_n \to \mathbb{Z}_n$ be defined by

$$\begin{aligned} E(x) &= a^x \bmod n \\ &= a \cdot_n a \cdot_n \ldots \cdot_n a, \end{aligned}$$

where we multiply x copies of a together. We see in Table 5 that for some values of a and n, the restriction of E to the non-zero integers is injective and for others it is not. Perhaps surprisingly, the last column of Table 5 consists of 1s only.

FERMAT'S LITTLE THEOREM. Let p be prime. Then $a^{p-1} \bmod p = 1$ for every non-zero $a \in \mathbb{Z}_p$.

PROOF. Since p is prime, multiplication with a gives an injective function for every non-zero $a \in \mathbb{Z}_p$. In other words, multiplying with a permutes the non-zero integers

a^x	0	1	2	3	4	5	6
1	1	1	1	1	1	1	1
2	1	2	4	1	2	4	1
3	1	3	2	6	4	5	1
4	1	4	2	1	4	2	1
5	1	5	4	6	2	3	1
6	1	6	1	6	1	6	1

Table 5: Exponentiation modulo $n = 7$. We write x from left to right and a from top to bottom.

in $\mathbb{Z}_p$. Hence,

$$\begin{aligned} X &= 1 \cdot_p 2 \cdot_p \ldots \cdot_p (p-1) \\ &= (1 \cdot_p a) \cdot_p (2 \cdot_p a) \cdot_p \ldots \cdot_p ((p-1) \cdot_p a) \\ &= X \cdot_p (a^{p-1} \bmod p). \end{aligned}$$

Multiplying with the inverse of X gives $a_{p-1} \bmod p = 1$. □

One-way functions. The RSA cryptosystem is based on the existence of *one-way functions* $f : \mathbb{Z}_n \to \mathbb{Z}_n$ defined by the following three properties:

- f is easy to compute;
- its inverse, $f^{-1} : \mathbb{Z}_n \to \mathbb{Z}_n$, exists;
- without extra information, f^{-1} is hard to compute.

The notions of 'easy' and 'hard' computation have to be made precise, but this is beyond the scope of this course. Roughly, it means that given x, computing $y = f(x)$ takes on the order of a few seconds while computing $f^{-1}(y)$ takes on the order of years. RSA uses the following recipe to construct one-way functions:

1. choose large primes p and q, and let $n = pq$;
2. choose $e \neq 1$ relative prime to $(p-1)(q-1)$ and let d be its multiplicative inverse modulo $(p-1)(q-1)$;
3. the one-way function is defined by $f(x) = x^e \bmod n$ and its inverse is defined by $g(y) = y^d \bmod n$.

According to the RSA protocol, Bob publishes e and n and keeps d private. To exchange a secret message, $x \in \mathbb{Z}_n$,

4. Alice computes $y = f(x)$ and publishes y;
5. Bob reads y and computes $z = g(y)$.

To show that RSA is secure, we would need to prove that without knowing p, q, d, it is hard to compute g. We

leave this to future generations of computer scientists. Indeed, nobody today can prove that computing p and q from $n = pq$ is hard, but then nobody knows how to factor large integers efficiently either.

Correctness. To show that RSA works, we need to prove that $z = x$. In other words, $g(y) = f^{-1}(y)$ for every $y \in \mathbb{Z}_n$. Recall that y is computed as $f(x) = x^e \bmod n$. We need $y^d \bmod n = x$ but we first prove a weaker result.

LEMMA. $y^d \bmod p = x \bmod p$ for every $x \in \mathbb{Z}_n$.

PROOF. Since d is the multiplicative inverse of e modulo $(p-1)(q-1)$, we can write $ed = (p-1)(q-1)k + 1$. Hence,

$$\begin{aligned} y^d \bmod p &= x^{ed} \bmod p \\ &= x^{k(p-1)(q-1)+1} \bmod p. \end{aligned}$$

Suppose first that $x^{k(q-1)} \bmod p \neq 0$. Then Fermat's Little Theorem implies $x^{k(p-1)(q-1)} \bmod p = 1$. But this implies $y^d \bmod p = x \bmod p$, as claimed. Suppose second that $x^{k(q-1)} \bmod p = 0$. Since p is prime, every power of a non-zero integer is non-zero. Hence, $x \bmod p = 0$. But this implies $y^d \bmod p = 0$ and thus $y^d \bmod p = x \bmod p$, as before. □

By symmetry, we also have $y^d \bmod q = x \bmod q$. Hence,

$$\begin{aligned} (y^d - x) \bmod p &= 0; \\ (y^d - x) \bmod q &= 0. \end{aligned}$$

By the Chinese Remainder Theorem, this system of two linear equations has a unique solution in $\mathbb{Z}_n$, where $n = pq$. Since $y^d - x = 0$ is a solution, there can be no other. Hence,

$$(y^d - x) \bmod n = 0.$$

The left hand side can be written as $((y^d \bmod n) - x) \bmod n$. This finally implies $y^d \bmod n = x$, as desired.

Summary. We talked about exponentiation modulo n and proved Fermat's Little Theorem. We then described how RSA uses exponentiation to construct one-way functions, and we proved it correct. A proof that RSA is secure would be nice but is beyond what is currently known.

Second Homework Assignment

Write the solution to each problem on a single page. The deadline for handing in solutions is February 6.

Question 1. ($20 = 10 + 10$ points). (Problem 2.1-12 in our textbook). We recall that a prime number, p, that divides a product of integers divides one of the two factors.

(a) Let $1 \leq a \leq p-1$. Use the above recollection to show that as b runs through the integers from 0 to $p-1$, the products $a \cdot_p b$ are all different.

(b) Explain why every positive integer less than p has a unique multiplicative inverse in $\mathbb{Z}_p$.

Question 2. (20 points). (Problem 2.2-19 in our textbook). The *least common multiple* of two positive integers i and j, denoted as $\mathrm{lcm}(i, j)$, is the smallest positive integer m such that m/i and m/j are both integer. Give a formula for $\mathrm{lcm}(i, j)$ that involves $\gcd(i, j)$.

Question 3. ($20 = 10 + 10$ points). (Problem 2.2-17 in our textbook). Recall the Fibonacci numbers defined by $F_0 = 0$, $F_1 = 1$, and $F_i = F_{i-1} + F_{i-2}$ for all $i \geq 2$.

(a) Run the extended gcd algorithm for $j = F_{10}$ and $k = F_{11}$, showing the values of all parameters at all levels of the recursion.

(b) Running the extended gcd algorithm for $j = F_i$ and $k = F_{i+1}$, how many recursive calls does it take to get the result?

Question 4. (20 points). Let $n \geq 1$ be a nonprime and $x \in \mathbb{Z}_n$ such that $\gcd(x, n) \neq 1$. Prove that $x^{n-1} \bmod n \neq 1$.

III Logic

It is now a good time to be more specific about the precise meaning of mathematical statements. They are governed by the rules of logic.

8 Boolean Algebra

Logic is generally considered to lie in the intersection between Philosophy and Mathematics. It studies the meaning of statements and the relationship between them.

Logical statements in computer programs. Programming languages provide all the tools to be excessively precise. This includes *logical statements* which are used to construct loops, among other things. As an example, consider a while loop that exchanges adjacent array elements until some condition expressed by a logical statement is satisfied. Putting the while loop inside a for loop we get a piece of code that sorts an array $A[1..n]$:

```
for i = 1 to n do j = i;
  while j > 1 and A[j] > A[j − 1] do
    a = A[j]; A[j] = A[j − 1]; A[j − 1] = a;
    j = j − 1
  endwhile
endfor.
```

This particular method for sorting is often referred to as insertion sort because after $i-1$ iterations, $A[1..i-1]$ is sorted, and the i-th iteration inserts the i-th element such that $A[1..i]$ is sorted. We illustrate the algorithm in Figure 7. Here we focus on the logic that controls the while loop.

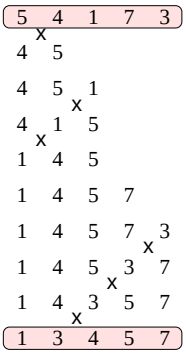


Figure 7: The insertion sort algorithm applied to an unsorted sequence of five integers.

The iteration is executed as long as two conditions hold, namely "$j > 1$" and "$A[j] > A[j-1]$". The first prevents we step beyond the left end of the array. The second condition limits the exchanges to cases in which adjacent elements are not yet in non-decreasing order. The two conditions are connected by a logical and, which requires both to be true.

Boolean operations. A logical statement is either true (T) of false (F). We call this the *truth value* of the statement. We will frequently represent the statement by a *variable* which can be either true or false. A *boolean operation* takes one or more truth values as input and produces a new output truth value. It thus functions very much like an arithmetic operation. For example, *negation* is a unary operation. It maps a truth value to the opposite; see Table 6. Much more common are binary operations; such as

p	$\neg p$
T	F
F	T

Table 6: Truth table for negation ($\neg$).

and, or, and exclusive or. We use a truth table to specify the values for all possible combinations of inputs; see Table 7. Binary operations have two input variables, each in one of two states. The number of different inputs is therefore only four. We have seen the use of these particular

p	q	$p \wedge q$	$p \vee q$	$p \oplus q$
T	T	T	T	F
T	F	F	T	T
F	T	F	T	T
F	F	F	F	F

Table 7: Truth table for and ($\wedge$), or ($\vee$), and exclusive or ($\oplus$) operations.

boolean operations before, namely in the definition of the common set operations; see Figure 8.

$$\begin{aligned} A^c &= \{x \mid x \notin A\}; \\ A \cap B &= \{x \mid x \in A \text{ and } x \in B\}; \\ A \cup B &= \{x \mid x \in A \text{ or } x \in B\}; \\ A \oplus B &= \{x \mid x \in A \text{ xor } x \in B\}; \\ A - B &= \{x \mid x \in A \text{ and } x \notin B\}. \end{aligned}$$

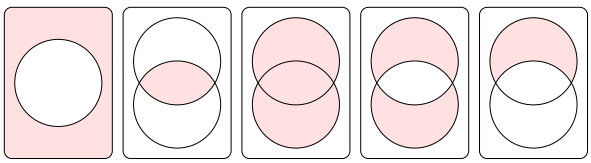


Figure 8: From left to right: the complement of one set and the intersection, union, symmetric difference, and difference of two sets.

Algebraic properties. We observe that boolean operations behave very much like ordinary arithmetic operations. For example, they follow the same kind of rules when we transform them.

- All three binary operations are commutative, that is,

$$\begin{aligned} p \wedge q \quad &\text{iff} \quad q \wedge p; \\ p \vee q \quad &\text{iff} \quad q \vee p; \\ p \oplus q \quad &\text{iff} \quad q \oplus p. \end{aligned}$$

- The and operation distributes over the or operation, and vice versa, that is,

$$\begin{aligned} p \wedge (q \vee r) \quad &\text{iff} \quad (p \wedge q) \vee (p \wedge r); \\ p \vee (q \wedge r) \quad &\text{iff} \quad (p \vee q) \wedge (p \vee r). \end{aligned}$$

Similarly, negation distributes over and and or, but it changes one into the other as it does so. This is known as de Morgan's Law.

DE MORGAN'S LAW. Letting p and q be two variables,

$$\begin{aligned} \neg(p \wedge q) \quad &\text{iff} \quad \neg p \vee \neg q; \\ \neg(p \vee q) \quad &\text{iff} \quad \neg p \wedge \neg q. \end{aligned}$$

PROOF. We construct the truth table, with a row for each combination of truth values for p and q; see Table 8. Since

p	q	$\neg(p \wedge q)$	$\neg p \vee \neg q$
T	T	F	F
T	F	T	T
F	T	T	T
F	F	T	T

Table 8: The truth table for the expressions on the left and the right of the first de Morgan Law.

the two relations are symmetric, we restrict our attention to the first. We see that the truth values of the two expressions are the same in each row, as required. □

Implications. The *implication* is another kind of binary boolean operation. It frequently occurs in statements of lemmas and theorems. An example is Fermat's Little Theorem. To emphasize the logical structure, we write A for the statement "n is prime" and B for "$a^{n-1} \bmod n = 1$ for every non-zero $a \in \mathbb{Z}_n$". There are different, equivalent ways to restate the theorem, namely "if A then B"; "A implies B"; "A only if B"; "B if A". The operation is

p	q	$p \Rightarrow q$	$\neg q \Rightarrow \neg p$	$\neg(p \wedge \neg q)$	$\neg p \vee q$
T	T	T	T	T	T
T	F	F	F	F	F
F	T	T	T	T	T
F	F	T	T	T	T

Table 9: The truth table for the implication ($\Rightarrow$).

defined in Table 9. We see the contrapositive in the second column on the right, which is equivalent, as expected. We also note that q is forced to be true if p is true and that q can be anything if p is false. This is expressed in the third column on the right, which relates to the last column by de Morgan's Law. The corresponding set operation is the complement of the difference, $(A - B)^c$; see Figure 9 on the left.

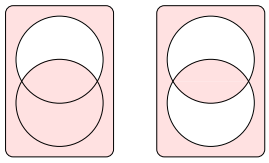


Figure 9: Left: the complement of the difference between the two sets. Right: the complement of the symmetric difference.

We recall that a logical statement is either true or false. This is referred to as the law of the *excluded middle*. In other words, a statement is true precisely when it is not false. There is no allowance for ambiguities or paradoxes. An example is the sometimes counter-intuitive definition that false implies true is true. Write A for the statement "it is raining", B for "I use my umbrella", and consider $A \Rightarrow B$. Hence, if it is raining then I use my umbrella. This does not preclude me from using the umbrella if it is not raining. In other words, the implication is not false if I use my umbrella without rain. Hence, it is true.

Equivalences. If implications go both ways, we have an *equivalence*. An example is the existence of a multiplicative inverse iff the multiplication permutes. We write A for the statement "a has a multiplicative inverse in $\mathbb{Z}_n$" and B for "the function $M : \mathbb{Z}_n \to \mathbb{Z}_n$ defined by $M(x) = a \cdot_n x$ is bijective". There are different, equivalent ways to restate the claim, namely "A if and only if B" and "A and B are equivalent". The operation is defined in Table 10. The last column shows that equivalence is the opposite of the exclusive or operation. Figure 9 shows the corresponding set operation on the right.

Recalling the definition of a group, we may ask which

p	q	$p \Leftrightarrow q$	$(p \Rightarrow q) \wedge (q \Rightarrow p)$	$\neg(p \oplus q)$
T	T	T	T	T
T	F	F	F	F
F	T	F	F	F
F	F	T	T	T

Table 10: The truth table for the equivalence ($\Leftrightarrow$).

of the binary operations form an Abelian group. The set is $\{\mathsf{F}, \mathsf{T}\}$. One of the two must be the neutral element. If we choose F then $\mathsf{F} \circ \mathsf{F} = \mathsf{F}$ and $\mathsf{F} \circ \mathsf{T} = \mathsf{T} \circ \mathsf{F} = \mathsf{T}$. Furthermore, $\mathsf{T} \circ \mathsf{T} = \mathsf{F}$ is necessary for T to have an inverse. We see that the answer is the exclusive or operation. Mapping F to 0 and T to 1, as it is commonly done in programming languages, we see that the exclusive or can be interpreted as adding modulo 2. Hence, $(\{\mathsf{F}, \mathsf{T}\}, \oplus)$ is isomorphc to $(\mathbb{Z}_2, +_2)$.

Summary. We have learned about the main components of logical statements, boolean variables and operations. We have seen that the operations are very similar to the more familiar arithmetic operations, mapping one or more boolean input variable to a boolean output variable.

9 Quantifiers

Logical statements usually include *variables*, which range over sets of possible instances, often referred to as *universes*. We use quantifiers to specify that something holds for all possible instances or for some but possibly not all instances.

Universal and existential quantifiers. We introduce the concept by taking an in-depth look at a result we have discussed in Chapter II.

EUCLID'S DIVISION THEOREM. Letting n be a positive integer, for every integer m there are unique integers q and r, with $0 \leq r < n$, such that $m = nq + r$.

In this statement, we have n, m, q, r as variables. They are integers, so $\mathbb{Z}$ is the universe, except that some of the variables are constrained further, that is, $n \geq 1$ and $0 \leq r < n$. The claim is "for all" m "there exist" q and r. These are quantifiers expressed in English language. The first is called the *universal quantifier*:

$\forall x \ [p(x)]$**:** for all instantiations of the variable x, the statement $p(x)$ is true.

For example, if x varies over the integers then this is equivalent to

$$\ldots \wedge p(-1) \wedge p(0) \wedge p(1) \wedge p(2) \wedge \ldots$$

The second is the *existential quantifier*:

$\exists x \ [q(x)]$**:** there exists an instantiation of the variable x such that the statement $q(x)$ is true.

For the integers, this is equivalent to

$$\ldots \vee q(-1) \vee q(0) \vee q(1) \vee q(2) \vee \ldots$$

With these quantifiers, we can restate Euclid's Division Theorem more formally:

$$\forall n \geq 1 \ \forall m \ \exists q \ \exists 0 \leq r < n [m = nq + r].$$

Negating quantified statements. Recall de Morgan's Law for negating a conjunction or a disjunction:

$$\begin{aligned} \neg(p \wedge q) &\quad \Leftrightarrow \quad \neg p \vee \neg q; \\ \neg(p \vee q) &\quad \Leftrightarrow \quad \neg p \wedge \neg q. \end{aligned}$$

The corresponding rules for quantified statements are

$$\begin{aligned} \neg\,(\forall x \ [p(x)]) &\quad \Leftrightarrow \quad \exists x \ [\neg p(x)]; \\ \neg\,(\exists x \ [q(x)]) &\quad \Leftrightarrow \quad \forall x \ [\neg q(x)]. \end{aligned}$$

We get the first line by applying de Morgan's first Law to the conjunction that corresponds to the expression on the left hand side. Similarly, we get the second line by applying de Morgan's second Law. Alternatively, we can derive the second line from the first. Since both sides of the first line are equivalent, so are its negations. Now, all we need to do it to substitute $\neg q(x)$ for $p(x)$ and exchange the two sides, which we can because $\Leftrightarrow$ is commutative.

Big-Oh notation. We practice the manipulation of quantified statements by discussing the big-Oh notation for functions. It is commonly used in statements about the convergence of an iteration or the running time of an algorithm. We write $\mathbb{R}^+$ for the set of positive real numbers.

DEFINITION. Let f and g be functions from $\mathbb{R}^+$ to $\mathbb{R}^+$. Then $f = \mathrm{O}(g)$ if there are positive constants c and n_0 such that $f(x) \leq cg(x)$ whenever $x > n_0$.

This notation is useful in comparing the asymptotic behavior of the functions f and g, that is, beyond a constant n_0. If $f = \mathrm{O}(g)$ then f can grow at most a constant times as fast as g. For example, we do not have $f = \mathrm{O}(g)$ if $f(x) = x^2$ and $g(x) = x$. Indeed, $f(x) = xg(x)$ so there is no constant c such that $f(x) \leq cg(x)$ because we can always choose x larger than c and n_0 and get a contradiction. We rewrite the definition in more formal notation. The statement $f = \mathrm{O}(g)$ is equivalent to

$$\exists c > 0 \ \exists n_0 > 0 \ \forall x \in \mathbb{R} \ [x > n_0 \Rightarrow f(x) \leq cg(x)].$$

We can simplify by absorbing the constraint of x being larger than the constant n_0 into the last quantifying statement:

$$\exists c > 0 \ \exists n_0 > 0 \ \forall x > n_0 \ [f(x) \leq cg(x)].$$

We have seen above that negating a quantified statement reverses all quantifiers and pulls the negation into the unquantified, inner statement. Recall that $\neg(p \Rightarrow q)$ is equivalent to $p \wedge \neg q$. Hence, the statement $f \neq \mathrm{O}(g)$ is equivalent to

$$\forall c > 0 \ \forall n_0 > 0 \ \exists x \in \mathbb{R} \ [x > n_0 \wedge f(x) > cg(x)].$$

We can again simplify by absorbing the constraint on x into the quantifying statement:

$$\forall c > 0 \ \forall n_0 > 0 \ \exists x > n_0 \ [f(x) > cg(x)].$$

Big-Theta notation. Recall that the big-Oh notation is used to express that one function grows asymptotically at most as fast as another, allowing for a constant factor of difference. The big-Theta notation is stronger and expresses that two functions grow asymptotically at the same speed, again allowing for a constant difference.

DEFINITION. Let f and g be functions from $\mathbb{R}^+$ to $\mathbb{R}^+$. Then $f = \Theta(g)$ if $f = \mathrm{O}(g)$ and $g = \mathrm{O}(f)$.

Note that in big-Oh notation, we can always increase the constants c and n_0 without changing the truth value of the statement. We can therefore rewrite the big-Theta statement using the larger of the two constants c and the larger of the two constants n_0. Hence, $f = \Theta(g)$ is equivalent to

$$\exists c > 0\, \exists n_0 > 0\, \forall x > n_0\ [f(x) \leq cg(x) \wedge g(x) \leq cf(x)].$$

Here we can further simplify by rewriting the two inequalities by a single one: $\frac{1}{c}g(x) \leq f(x) \leq cg(x)$. Just for practice, we also write the negation in formal notation. The statement $f \neq \Theta(f)$ is equivalent to

$$\forall c > 0\, \forall n_0 > 0\, \exists x > n_0\ [cg(x) < f(x) \vee cf(x) < g(x)].$$

Because the two inequalities are connected by a logical or, we cannot simply combine them. We could by negating it first, $\neg(\frac{1}{c}g(x) \leq f(x) \leq cg(x))$, but this is hardly easier to read.

Big-Omega notation. Complementary to the big-Oh notation, we have

DEFINITION. Let f and g be functions from $\mathbb{R}^+$ to $\mathbb{R}^+$. Then $f = \Omega(g)$ if $g = \mathrm{O}(f)$.

In formal notation, $f = \Omega(g)$ is equivalent to

$$\exists c > 0\, \exists n_0 > 0 \forall x > n_0\ [f(x) \geq cg(x)].$$

We may think of big-Oh like a less-than-or-equal-to for functions, and big-Omega as the complementary greater-than-or-equal-to. Just as we have $x = y$ iff $x \leq y$ and $x \geq y$, we have $f = \Theta(g)$ iff $f = \mathrm{O}(g)$ and $f = \Omega(g)$.

Little-oh and little-omega notation. For completeness, we add notation that corresponds to the strict less-than and greater-than relations.

DEFINITION. Let f and g be functions from $\mathbb{R}^+$ to $\mathbb{R}^+$. Then $f = \mathrm{o}(g)$ if for all constants $c > 0$ there exists a constant $n_0 > 0$ such that $f(x) < cg(x)$ whenever $x > n_0$. Furthermore, $f = \omega(g)$ if $g = \mathrm{o}(f)$.

This is not equivalent to $f = \mathrm{O}(g)$ and $f \neq \Omega(g)$. The reason for this is the existence of functions that cannot be compared at all. Consider for example $f(x) = x^2(\cos x + 1)$. For $x = 2k\pi$, k a non-negative integer, we have $f(x) = 2x^2$, while for $x = (2k+1)\pi$, we have $f(x) = 0$. Let $g(x) = x$. For even multiples of π, f grows much fast than g, while for odd multiples of π it grows much slower than g, namely not at all. We rewrite the little-Oh notation in formal notation. Specifically, $f = \mathrm{o}(g)$ is equivalent to

$$\forall c > 0\, \exists n_0 > 0\, \forall x > n_0\ [f(x) < cg(x)].$$

Similarly, $f = \omega(g)$ is equivalent to

$$\forall c > 0\, \exists n_0 > 0\, \forall x > n_0\ [f(x) > \frac{1}{c}g(x)].$$

In words, no matter how small our positive constant c is, there always exists a constant n_0 such that beyond that constant, $f(x)$ is larger than $g(x)$ over c. Equivalently, no matter how big our constant c is, there always exists a constant n_0 such that beyond that constant, $f(x)$ is larger than c times $g(x)$. We can thus simplify the formal statement by substituting $[f(x) > cg(x)]$ for the inequality.

10 Inference

In this section, we discuss the application of logic to proving theorems. In principle, every proof should be reducible to a sequence of simple logical deductions. While this is not practical for human consumption, there have been major strides toward that goal in computerized proof systems.

Modus ponens. This is an example of *direct inference*, the cornerstone of logical arguments.

PRINCIPLE OF MODUS PONENS. From p and $p \Rightarrow q$, we may conclude q.

We read this as a recipe to prove q. First we prove p, then we prove that p implies q, and finally we conclude q. Let us take a look at Table 11 to be sure. We see that modus

p	q	$(p \wedge (p \Rightarrow q))$	$\Rightarrow$	q
T	T	T T T	T	T
T	F	T F F	T	F
F	T	F F T	T	T
F	F	F F T	T	F

Table 11: The truth table for modus ponens.

ponens is indeed a tautology, that is, it is always true. Every theorem is this way, namely always true.

Other methods of direct inference. There are many other direct proof principles, all easy to verify. Some are straighforward re-interpretations of logical formulas and others use logical equivalences we have learned about. Here are but a few:

$$\begin{aligned} p \text{ and } q \quad &\text{then} \quad p \wedge q; \\ p \text{ or } q \quad &\text{then} \quad p \vee q; \\ q \text{ or } \neg p \quad &\text{then} \quad p \Rightarrow q; \\ \neg q \text{ and } p \quad &\text{then} \quad p \not\Rightarrow q; \\ p \Rightarrow q \text{ and } q \Rightarrow p \quad &\text{then} \quad p \Leftrightarrow q; \\ p \Rightarrow q \text{ and } q \Rightarrow r \quad &\text{then} \quad p \Leftrightarrow r. \end{aligned}$$

The last principle is perhaps more interesting than the others because it is the only one among the six that is not an equivalence; see Table 12.

Contrapositive. This is the first example of an *indirect inference* method.

p	q	r	$((p \Rightarrow q) \wedge (q \Rightarrow r))$	$\Rightarrow$	$(p \Rightarrow r)$
T	T	T	T T T	T	T
T	T	F	T F F	T	F
T	F	T	F F T	T	T
T	F	F	F F T	T	F
F	T	T	T T T	T	T
F	T	F	T F F	T	T
F	F	T	T T T	T	T
F	F	F	T T T	T	T

Table 12: The truth table for reasoning by transitivity.

PRINCIPLE OF CONTRAPOSITION. The statements $p \Rightarrow q$ and $\neg q \Rightarrow \neg p$ are equivalent, and so a proof of one is a proof of the other.

We have seen a truth table that shows the equivalence of the two statements earlier, in Section 8. Let us look at an example.

CLAIM. If n is a positive integer with $n^2 > 25$ then $n > 5$.

PROOF. The statement p is that n is a positive integer whose square is larger than 25. The statement q is that n is larger than 5. We could argue directly but then we would need to know something about talking square roots. Instead, let us argue indirectly. Suppose $\neg q$, that is, $n \leq 5$. By monotonicity of multiplication, we have

$$n^2 \;\leq\; 5n \;\leq\; 5 \cdot 5 \;\leq\; 25.$$

Now, by transitivity of the smaller-than-or-equal-to relation, we have $n^2 \leq 25$. Thus $\neg q$ implies $\neg p$. □

Example: Chinese remainders. Another instructive example is a result we have seen in Section 6. Let m and n be relative prime, positive integers. We map each integer in $\mathbb{Z}_{mn}$ to the pair of remainders, that is, for $0 \leq x < mn$ we define $f(x) = (x \bmod m, x \bmod n)$.

CHINESE REMAINDER THEOREM. If $x \neq y$ both belong to $\mathbb{Z}_{mn}$ then $f(x) \neq f(y)$.

PROOF. We use again the indirect approach by contraposition. Assume $f(x) = f(y)$. Then

$$\begin{aligned} x \bmod m &= y \bmod m; \\ x \bmod n &= y \bmod n. \end{aligned}$$

Hence,

$$\begin{aligned}(x-y) \bmod m &= 0;\\ (x-y) \bmod n &= 0.\end{aligned}$$

Therefore, $x-y$ is a multiple of both m and n. Hence, $(x-y) \bmod mn = 0$ and therefore $x \bmod mn = y \bmod mn$, which contradicts that $x \neq y$ in $\mathbb{Z}_{mn}$. □

Reduction to Absurdity. Another powerful indirect proof technique is by contradiction.

PRINCIPLE OF REDUCTION TO ABSURDITY. If from assuming p and $\neg q$ we can derive r as well as $\neg r$ then $p \Rightarrow q$.

Here r can be any statement. Often we use a statement r that is always true (or always false) so that we only need to derive $\neg r$ (or r) from p and $\neg q$. Let us take a look at Table 13. As with all the proof methods, it is best to see examples. There are many and a large variety because different principles are combined, or made more complicated, etc.

p	q	r	$((p \wedge \neg q) \Rightarrow (r \wedge \neg r))$	$\Rightarrow$	$(p \Rightarrow q)$
T	T	T	F T F	T	T
T	T	F	F T F	T	T
T	F	T	T F F	T	F
T	F	F	T F F	T	F
F	T	T	F T F	T	T
F	T	F	F T F	T	T
F	F	T	F T F	T	T
F	F	F	F T F	T	T

Table 13: The truth table for the reduction to absurdity.

Example: irrational numbers. A real number u is *rational* if there are integers m and n such that $u = \frac{m}{n}$ and *irrational* otherwise. The set of rational numbers is denoted as $\mathbb{Q}$. For any two different rational numbers, $u < w$, we can always find a third that lies strictly between them. For example, if $w = \frac{k}{l}$ then

$$\begin{aligned}v &= \frac{u+w}{2}\\ &= \frac{ml+nk}{nl}\end{aligned}$$

lies halfway between u and w. This property is sometimes expressed by saying the rational numbers are *dense* in the set of real numbers. How do we know that not all real numbers are rational?

CLAIM. $\sqrt{5}$ is irrational.

PROOF. Assume the square root of 5 is rational, that is, there exist integers m and n such that $\sqrt{5} = \frac{m}{n}$. Squaring the two sides, we get

$$5 = \frac{m^2}{n^2}$$

or, equivalently, $5n^2 = m^2$. But m^2 has an even number of prime factors, namely each factor twice, while $5n^2$ has an odd number of prime factors, namely 5 together with an even number of prime factors for n^2. Hence, $5n^2 = m^2$ is not possible, a contradiction. □

We take a look at the logic structure of this proof. Let p be the statement that $\sqrt{5}^2 = 5$ and q the statement that $\sqrt{5}$ is irrational. Thus $\neg q$ is the statement that $\sqrt{5} = \frac{m}{n}$. From assuming p and $\neg q$, we derive r, that is the statement $5n^2 = m^2$. But we also have $\neg r$, because each integer has a unique decomposition into prime factors. We thus derived r and $\neg r$. But this cannot be true. Using the Principle of Reduction to Absurdity, we conclude that p implies q. By modus ponens, assuming p gives q.

Summary. We have learned that theorems are tautologies and there are different ways to prove them. As applications of logic rules we have discussed direct methods (Principle of Modus Ponens) and indirect methods (Principle of Contrapositive and Principle of Reduction to Absurdity).

Third Homework Assignment

Write the solution to each problem on a single page. The deadline for handing in solutions is February 23.

Question 1. ($20 = 10 + 10$ points). (Problem 3.1-6 in our textbook). Show that $p \oplus q$ is equivalent to $(p \wedge \neg q) \vee (\neg p \wedge q)$. State the corresponding relation in terms of sets and set operations.

Question 2. ($20 = 10 + 10$ points). (Problem 3.2-14 in our textbook). Let x, y, z be variables and p, q logical statements that depend on one variable.

(a) Are the following two compound logical statements equivalent?

1. $(\exists x \in \mathbb{R}\ [p(x)]) \wedge (\exists y \in \mathbb{R}\ [q(y)])$;
2. $\exists z \in \mathbb{R}\ [p(z) \wedge q(z)]$.

(Justify your answer.)

(b) Are the following two compound logical statements equivalent?

1. $(\exists x \in \mathbb{R}\ [p(x)]) \vee (\exists y \in \mathbb{R}\ [q(y)])$;
2. $\exists z \in \mathbb{R}\ [p(z) \vee q(z)]$.

(Justify your answer.)

Question 3. (20 points). (Problem 3.3-6 in our textbook). Is the statement $p \Rightarrow q$ equivalent to the statement $\neg p \Rightarrow \neg q$? (If yes, why? If no, why not?)

Question 4. (20 points). (Problem 3.3-14 in our textbook). Prove that there is no largest prime number. In other words, for every prime number there is another, larger prime number.

IV INDUCTION

This is a general purpose proof technique that works in a bottom-up fashion. Knowing that a statement is true for a collection of instances, we argue that it is also true for a new instance, which we then add to the collection. Repeating this step, we establish the statement for a countable collection.

11 Mathematical Induction

In philosophy, *deduction* is the process of taking a general statement and applying it to a specific instance. For example: all students must do homework, and I am a student; therefore, I must do homework. In contrast, *induction* is the process of creating a general statement from observations. For example: all cars I have owned need to be repaired at some point; therefore, all cars will need to be repaired at some point. A similar concept is used in mathematics to prove that a statement is true for all integers. To distinguish it from the less specific philosophical notion, we call it *mathematical induction* of which we will introduce two forms. We begin by considering an example from Section 4, showing that the idea behind Mathematical Induction is a familiar one.

Euclid's Division Theorem. We find the smallest counterexample in order to prove the following theorem.

EUCLID'S DIVISION THEOREM. Letting $n \geq 1$, for every non-negative integer m there are unique integers q and $0 \leq r < n$ such that $m = nq + r$.

PROOF. Assume the opposite, that is, there is a non-negative integer m for which no such q and r exist. We choose the smallest such m. Note that m cannot be smaller than n, else we have $q = 0$ and $r = m$, and m cannot be equal to n, else we have $q = 1$ and $r = 0$. It follows that $m' = m - n$ is a positive integer less than m. Thus, there exist integers q' and $0 \leq r' < n$ such that $m' = nq' + r'$. If we add n on both sides, we obtain $m = (q' + 1)n + r'$. If we take $q = q' + 1$ and $r = r'$, we get $m = nq + r$, with $0 \leq r < n$. Thus, by the Principle of Reduction to Absurdity, such integers q and r exist. □

Let $p(k)$ be the statement that there exist integers q and $0 \leq r < n$ with $k = nq + r$. Then, the above proof can be summarized by

$$p(m-n) \wedge \neg p(m) \implies p(m) \wedge \neg p(m).$$

This is the contradiction that implies $\neg p(m)$ cannot be true. We now focus on the statement $p(m-n) \Rightarrow p(m)$. This is the idea of Mathematical Induction which bypasses the construction of a contradiction.

Example: sum of integers. We consider the familiar problem of summing the first n positive integers. Recall that $\binom{n+1}{2} = \frac{n(n+1)}{2}$.

CLAIM. For all $n \geq 0$, we have $\sum_{i=0}^{n} i = \binom{n+1}{2}$.

PROOF. First, we note that $\sum_{i=0}^{0} i = 0 = \binom{1}{2}$. Now, we assume inductively that for $n > 0$, we have

$$\sum_{i=0}^{n-1} i = \binom{n}{2}.$$

If we add n on both sides, we obtain

$$\begin{aligned} \sum_{i=0}^{n} i &= \binom{n}{2} + n \\ &= \frac{(n-1)n}{2} + \frac{2n}{2} \end{aligned}$$

which is $\frac{(n+1)n}{2} = \binom{n+1}{2}$. Thus, by the Principle of Mathematical Induction,

$$\sum_{i=0}^{n} i = \binom{n+1}{2}$$

for all non-negative integers n. □

To analyze why this proof is correct, we let $p(k)$ be the statement that the claim is true for $n = k$. For $n = 1$ we have $p(1) \wedge [p(1) \Rightarrow p(2)]$. Hence, we get $p(2)$ by Modus Ponens. We can see that this continues:

$$\begin{array}{rlll} p(1) \wedge [p(1) \Rightarrow p(2)] & \text{hence} & p(2); \\ p(2) \wedge [p(2) \Rightarrow p(3)] & \text{hence} & p(3); \\ \cdots & \cdots & \cdots \\ p(n-1) \wedge [p(n-1) \Rightarrow p(n)] & \text{hence} & p(n); \\ \cdots & \cdots & \cdots \end{array}$$

Thus, $p(n_0)$ and $p(n-1) \Rightarrow p(n)$ for all $n > n_0$ implies $p(n)$ for all $n \geq n_0$.

The weak form. We formalize the proof technique into the first, weak form of the principle. The vast majority of applications of Mathematical Induction use this particular form.

MATHEMATICAL INDUCTION (WEAK FORM). If the statement $p(n_0)$ is true, and the statement $p(n-1) \Rightarrow p(n)$ is true for all $n > n_0$, then $p(n)$ is true for all integers $n \geq n_0$.

To write a proof using the weak form of Mathematical Induction, we thus take the following four steps: it should have the following components:

Base Case: $p(n_0)$ is true.

Inductive Hypothesis: $p(n-1)$ is true.

Inductive Step: $p(n-1) \Rightarrow p(n)$.

Inductive Conclusion: $p(n)$ for all $n \geq n_0$.

Very often but not always, the inductive step is the most difficult part of the proof. In practice, we usually sketch the inductive proof, only spelling out the portions that are not obvious.

Example: sum of powers of two. If we can guess the closed form expression for a finite sum, it is often easy to use induction to prove that it is correct, if it is.

CLAIM. For all integers $n \geq 1$, we have $\sum_{i=1}^{n} 2^{i-1} = 2^n - 1$.

PROOF. We prove the claim by the weak form of the Principle of Mathematical Induction. We observe that the equality holds when $n = 1$ because $\sum_{i=1}^{1} 2^{i-1} = 1 = 2^1 - 1$. Assume inductively that the claim holds for $n-1$. We get to n by adding 2^{n-1} on both sides:

$$\begin{aligned} \sum_{i=1}^{n} 2^{i-1} &= \sum_{i=1}^{n-1} 2^{i-1} + 2^{n-1} \\ &= (2^{n-1} - 1) + 2^{n-1} \\ &= 2^n - 1. \end{aligned}$$

Here, we use the inductive assumption to go from the first to the second line. Thus, by the Principle of Mathematical Induction, $\sum_{i=1}^{n} 2^{i-1} = 2^n - 1$ for all $n \geq 1$. □

The strong form. Sometimes it is not enough to use the validity of $p(n-1)$ to derive $p(n)$. Indeed, we have $p(n-2)$ available and $p(n-3)$ and so on. Why not use them?

MATHEMATICAL INDUCTION (STRONG FORM). If the statement $p(n_0)$ is true and the statement $p(n_0) \wedge p(n_0+1) \wedge \cdots \wedge p(n-1) \Rightarrow p(n)$ is true for all $n > n_0$, then $p(n)$ is true for all integers $n \geq n_0$.

Notice that the strong form of the Principle of Mathematical Induction implies the weak form.

Example: prime factor decomposition. We use the strong form to prove that every integer has a decomposition into prime factors.

CLAIM. Every integer $n \geq 2$ is the product of prime numbers.

PROOF. We know that 2 is a prime number and thus also a product of prime numbers. Suppose now that we know that every positive number less than n is a product of prime numbers. Then, if n is a prime number we are done. Otherwise, n is not a prime number. By definition of prime number, we can write it is the product of two smaller positive integers, $n = a \cdot b$. By our supposition, both a and b are products of prime numbers. The product, $a \cdot b$, is obtained by merging the two products, which is again a product of prime numbers. Therefore, by the strong form of the Principle of Mathematical Induction, every integer $n \geq 2$ is a product of prime numbers. □

We have used an even stronger statement before, namely that the decomposition into prime factors is unique. We can use the Reduction to Absurdity to prove uniqueness. Suppose n is the smallest positive integer that has two different decompositions. Let $a \geq 2$ be the smallest prime factor in the two decompositions. It does not belong to the other decomposition, else we could cancel the two occurrences of a and get a smaller integer with two different decompositions. Clearly, $n \bmod a = 0$. Furthermore, $r_i = b_i \mod a \neq 0$ for each prime factor b_i in the other decomposition of n. We have

$$\begin{aligned} n \bmod a &= \left(\prod_i b_i\right) \bmod a \\ &= \left(\prod_i r_i\right) \bmod a. \end{aligned}$$

Since all the r_i are smaller than a and a is a prime number, the latter product can only be zero if one or the r_i is zero. But this contradicts that all the b_i are prime numbers larger than a. We thus conclude that every integer larger than one has a unique decomposition into prime factors.

Summary. Mathematical Induction is a tool to prove that a property is true for all positive integers. We used Modus Ponens to prove the weak as well as the strong form of the Principle of Mathematical Induction.

12 Recursion

We now describe how recurrence relations arise from recursive algorithms, and begin to look at ways of solving them. We have just learned one method that can sometimes be used to solve such a relation, namely Mathematical Induction. In fact, we can think of recursion as backwards induction.

The towers of Hanoi. Recurrence relations naturally arise in the analysis of the towers of Hanoi problem. Here we have three pegs, A, B, C, and initially n disks at A, sorted from large to small; see Figure 10. The task is to move the n disks from A to C, one by one, without ever placing a larger disk onto a smaller disk. The following

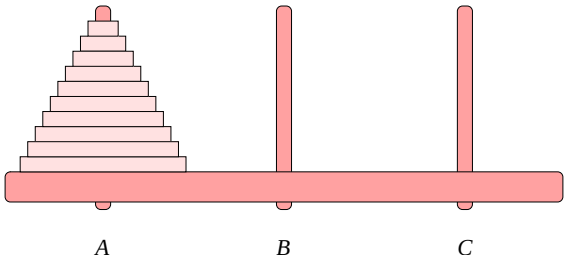


Figure 10: We have a sorted stack of disks at A and use B for temporary storage to move one disk at a time to C. We need B to avoid any inversions among the disks.

three steps solve this problem:

- recursively move $n-1$ disks from A to B;
- move the n-th disk from A to C;
- recursively move $n-1$ disks from B to C.

When we move disks from one peg to another, we use the third peg to help. For the main task, we use B to help. For the first step, we exchange the roles of B and C, and for the third step, we exchange the roles of A and B. The number of moves is given by the solution to the recurrence relation

$$M(n) \quad = \quad 2M(n-1)+1,$$

with initial condition $M(0)=0$. We may use induction to show that $M(n)=2^n-1$.

Loan payments. Another example in which recurrence relations naturally arise is the repayment of loans. This is an iterative process in which we alternate the payment of a constant sum with the accumulation of interest. The iteration ends when the entire loan is payed off. Suppose A_0 is the initial amount of your loan, m is your monthly payment, and p is the annual interest payment rate. What is the amount you owe after n months? We can express it in terms of the amount owed after $n-1$ months:

$$T(n) \quad = \quad \left(1+\frac{p}{12}\right)T(n-1)-m.$$

This is a recurrence relation, and figuring out how much you owe is the same as solving the recurrence relation. The number that we are most interested in is $n_0 m$, where n_0 is the number of months it takes to get $T(n_0)=0$. Instead of attacking this question directly, let us look at a more abstract, mathematical setting.

Iterating the recursion. Consider the following recurrence relation,

$$T(n) \quad = \quad rT(n-1)+a,$$

where r and a are some fixed real numbers. For example, we could set $r=1+\frac{p}{12}$ and $a=-m$ to get the recurrence that describes how much money you owe. After replacing $T(n)$ by $rT(n-1)+a$, we may take another step and replace $T(n-1)$ by $rT(n-2)+a$ to get $T(n)=r(rT(n-2)+a)+a$. Iterating like this, we get

$$\begin{aligned} T(n) &= rT(n-1)+a \\ &= r^2T(n-2)+ra+a \\ &= r^3T(n-3)+r^2a+ra+a \\ \cdots &\quad \cdots \\ &= r^nT(0)+a\sum_{i=0}^{n-1} r^i. \end{aligned}$$

The first term on the right hand side is easy, namely r^n times the initial condition, say $T(0)=b$. The second term is a sum, which we now turn into a nicer form.

Geometric series. The sequence of terms inside a sum of the form $\sum_{i=0}^{n-1} r^i$ is referred to as a *geometric series*. If $r=1$ then this sum is equal to n. To find a similarly easy expression for other values of r, we expand both the sum and its r-fold multiple:

$$\begin{aligned} \sum_{i=0}^{n-1} r^i &= r^0+r^1+r^2+\ldots+r^{n-1}; \\ r\sum_{i=0}^{n-1} r^i &= r^1+r^2+\ldots+r^{n-1}+r^n. \end{aligned}$$

Subtracting the second line from the first, we get

$$(1-r)\sum_{i=0}^{n-1} r^i \quad = \quad r^0 - r^n$$

and therefore $\sum_{i=0}^{n-1} r^i = \frac{1-r^n}{1-r}$. Now, this allows us to rewrite the solution to the recurrence as

$$T(n) \quad = \quad r^n b + a\frac{1-r^n}{1-r},$$

where $b = T(0)$ and $r \neq 1$. Let us consider the possible scenarios:

`Case 1.` $r = 0$. Then, $T(n) = a$ for all n.

`Case 2.` $0 < r < 1$. Then, $\lim_{n\to\infty} r^n = 0$. Therefore, $\lim_{n\to\infty} T(n) = \frac{a}{1-r}$.

`Case 3.` $r > 1$. The factors r^n of b and $\frac{r^n-1}{r-1}$ of a both grow with growing n. For positive values of a and b, we can expect $T(n) = 0$ for a negative value of n. Multiplying with $r-1$, we get $r^n b(r-1)+ar^n-a = 0$ or, equivalently, $r^n(br - b + a) = a$. Dividing by $br - b + a$, we get $r^n = \frac{a}{br-b+a}$, and taking the logarithm to the base r, we get

$$n \quad = \quad \log_r\left(\frac{a}{br-b+a}\right).$$

For positive values of a and b, we take the logarithm of a positive number smaller than one. The solution is a negative number n.

We note that the loan example falls into Case 3, with $r = 1 + \frac{p}{12} > 1$, $b = A_0$, and $a = -m$. Hence, we are now in a position to find out after how many months it takes to pay back the loan, namely

$$n_0 \quad = \quad \log_r\left(\frac{m}{m - A_0\frac{p}{12}}\right).$$

This number is well defined as long as $m > A_0\frac{p}{12}$, which means your monthly payment should exceed the monthly interest payment. It better happen, else the amount you owe grows and the day in which the loan will be payed off will never arrive.

First-order linear recurrences. The above is an example of a more general class of recurrence relations, namely the *first-order linear recurrences* that are of the form

$$T(n) \quad = \quad f(n)T(n-1) + g(n).$$

For the constant function $f(n) = r$, we have

$$\begin{aligned} T(n) \quad &= \quad r^n T(0) + \sum_{i=0}^{n-1} r^i g(n-i) \\ &= \quad r^n T(0) + \sum_{i=0}^{n-1} r^{n-i} g(i). \end{aligned}$$

We see that if $g(n) = a$, then we have the recurrence we used above. We consider the example $T(n) = 2T(n-1) + n$ in which $r = 2$ and $g(i) = i$. Hence,

$$\begin{aligned} T(n) \quad &= \quad 2^n T(0) + \sum_{i=0}^{n-1} \frac{i}{2^{n-i}} \\ &= \quad 2^n T(0) + \frac{1}{2^n}\sum_{i=0}^{n-1} i2^i. \end{aligned}$$

It is not difficult to find a closed form expression for the sum. Indeed, it is the special case for $x = 2$ of the following result.

CLAIM. For $x \neq 1$, we have

$$\sum_{i=1}^{n} ix^i \quad = \quad \frac{nx^{n+2} - (n-1)x^{n+1} + x}{(1-x)^2}.$$

PROOF. One way to prove the relation is by induction. Writing $R(n)$ for the right hand side of the relation, we have $R(1) = x$, which shows that the claimed relation holds for $n = 1$. To make the step from $n-1$ to n, we need to show that $R(n-1) + x^n = R(n)$. It takes but a few algebraic manipulations to show that this is indeed the case. □

Summary. Today, we introduced recurrence relations. To find the solution, we often have to define $T(n)$ in terms of $T(n_0)$ rather than $T(n-1)$. We also saw that different recurrences can have the same general form. Knowing this will help us to solve new recurrences that are similar to others that we have already seen.

13 Growth Rates

How does the time to iterate through a recursive algorithm grow with the size of the input? We answer this question for two algorithms, one for searching and the other for sorting. In both case, we find the answer by solving a recurrence relation.

Binary Search. We begin by considering a familiar algorithm, binary search. Suppose we have a sorted array, $A[1..n]$, and we wish to find a particular item, x. Starting in the middle, we ask whether $x = A[(n+1)/2]$? If it is, we are done. If not, we have cut the problem in half. We give a more detailed description in pseudo-code.

```
l = 1; r = n;
while l ≤ r do m = (l + r)/2;
  if x = A[m] then print(m); exit
    elseif x < A[m] then r = m − 1;
    elseif x > A[m] then l = m + 1
  endif
endwhile.
```

Assuming $n = 2^k - 1$, there are $2^{k-1} - 1$ items to the left and to the right of the middle. Let $T(n)$ be the number of times we check whether $l \leq r$. We check once at the beginning, for $n = 2^k - 1$ items, and then some number of times for half the items. In total, we have

$$T(n) = \begin{cases} T(\frac{n-1}{2}) + 1 & \text{if } n \geq 2; \\ 1 & \text{if } n = 1. \end{cases}$$

In each iteration, k decreases by one and we get $T(n) = k+1$. Since $k = \log_2(n+1)$, this gives $T(n) = 1+\log_2 n$. We could verify this by induction.

A similar recurrence relation. Let us consider another example, without specific algorithm. Suppose we solve a problem of size n by first solving one problem of size $n/2$ and then doing n units of additional work. Assuming n is a power of 2, we get the following recurrence relation:

$$T(n) = \begin{cases} T(\frac{n}{2}) + n & \text{if } n \geq 2; \\ 0 & \text{if } n = 1. \end{cases} \quad (1)$$

Figure 11 visualizes the computation by drawing a node for each level of the recursion. Even though the sequence of nodes forms a path, we call this the *recursion tree* of the computation. The problem size decreases by a factor of two from one level to the next. After dividing $\log_2 n$ times, we arrive at size one. This implies that there are only $1 + \log_2 n$ levels. Similarly, the work at each level decreases by a factor of two from one level to the next. Assuming $n = 2^k$, we get

$$\begin{aligned} T(n) &= n + \frac{n}{2} + \ldots + 2 + 1 \\ &= 2^k + 2^{k-1} + \ldots + 2^1 + 2^0 \\ &= 2^{k+1} - 1. \end{aligned}$$

Hence, $T(n) = 2n - 1$.

level	#nodes	size		work per node	work per level
1	1	n		n	n
2	1	$n/2$		$n/2$	$n/2$
3	1	$n/4$		$n/4$	$n/4$
4	1	$n/8$		$n/8$	$n/8$

Figure 11: The recursion tree for the relation in Equation (1).

Merge Sort. Next, we consider the problem of sorting a list of n items. We assume the items are stored in unsorted order in an array $A[1..n]$. The list is sorted if it consists of only one item. If there are two or more items then we sort the first $n/2$ items and the last $n/2$ items and finally merge the two sorted lists. We provide the pseudo-code below. We call the function with $\ell = 1$ and $r = n$.

```
void MergeSort(ℓ, r)
  if ℓ < r then m = (ℓ + r)/2;
    MergeSort(ℓ, m);
    MergeSort(m + 1, r);
    Merge(ℓ, m, r)
  endif.
```

We merge the two sorted lists by scanning them from left to right, using n comparisons. It is convenient to relocate both lists from A to another array, B, and to add a so-called stopper after each sublist. These are items that are larger than all given items. In other words, we assume the two lists are stored in $B[\ell..m]$ and $B[m+2..r+1]$, with $B[m+1] = B[r+2] = \infty$. When we scan the two lists, we move the items back to A, one at a time.

```
void MERGE(ℓ, m, r)
  i = ℓ; j = m + 2;
  for k = ℓ to r do
    if B[i] < B[j] then A[k] = B[i]; i = i + 1;
                   else A[k] = B[j]; j = j + 1
    endif
  endfor.
```

Assume $n = 2^k$ so that the sublists are always of the same length. The total number of comparisons is then

$$T(n) = \begin{cases} 2T(\frac{n}{2}) + n & \text{if } n \geq 2; \\ 0 & \text{if } n = 1. \end{cases}$$

To analyze this recurrence, we look at its recursion tree.

Recursion Tree. We begin with a list of length n, from which we create two shorter lists of length $n/2$ each. After sorting the shorter lists recursively, we use n comparisons to merge them. In Figure 12, we show how much work is done on the first four levels of the recursion. In this example, there are n units of work per level, and $1 + \log_2 n$ levels in the tree. Thus, sorting with the merge sort algorithm takes $T(n) = n \log_2 n + n$ comparisons. You can verify this using Mathematical Induction.

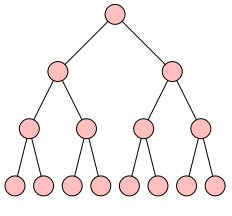
level	#nodes	size		work per node	work per level
1	1	n		n	n
2	2	$n/2$		$n/2$	n
3	4	$n/4$		$n/4$	n
4	8	$n/8$		$n/8$	n

Figure 12: The recursion tree for the merge sort algorithm.

Unifying the findings. We have seen several examples today, and we now generalize what we have found.

CLAIM. Let $a \geq 1$ be an integer and d a non-negative real number. Let $T(n)$ be defined for integers that are powers of 2 by

$$T(n) = \begin{cases} aT(\frac{n}{2}) + n & \text{if } n \geq 2; \\ d & \text{if } n = 1. \end{cases}$$

Then we have the following:

- $T(n) = \Theta(n)$ if $a < 2$;
- $T(n) = \Theta(n \log n)$ if $a = 2$;
- $T(n) = \Theta(n^{\log_2 a})$ if $a > 2$.

In the next lecture, we will generalize this result further so it includes our finding that binary search takes only a logarithmic number of comparisons. We will also see a justification of the three cases.

Summary. Today, we looked at growth rates. We saw that binary search grows logarithmically with respect to the input size, and merge sort grows at a rate of order $n \log_2 n$. We also discovered a pattern in a class recurrence relations.

14 Solving Recurrence Relations

Solving recurrence relations is a difficult business and there is no catch all method. However, many relations arising in practice are simple and can be solved with moderate effort.

A few functions. A solution to a recurrence relation is generally given in terms of a function, eg. $f(n) = n \log_2 n$, or a class of similar functions, eg. $T(n) = \mathrm{O}(n \log_2 n)$. It is therefore useful to get a feeling for some of the most common functions that occur. By plotting the graphs, as in Figure 13, we get an initial picture. Here we see a sequence of progressively faster growing functions: constant, logarithmic, linear, and exponential. However,

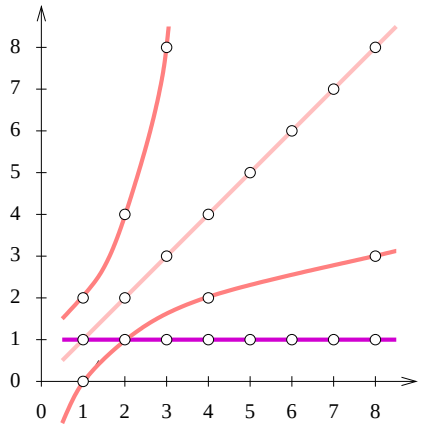


Figure 13: The graphs of a small set of functions, $f(x) = 1$, $f(x) = \log_2 x$, $f(x) = x$, $f(x) = 2^x$.

such plots can be confusing because they depend on the scale. For example, the exponential function, $f(x) = 2^x$, grows a lot faster than the quadratic function, $f(x) = x^2$, but this would not be obvious if we only look at a small portion of the plane like in Figure 13.

Three regimes. In a recurrence relation, we distinguish between the *homogeneous* part, the recursive terms, and the *inhomogeneous* part, the work that occurs. The solution of depends on the relative size of the two, exhibiting qualitatively different behavior if one dominates the other or the two are in balance. Recurrence relations that exhibit this three-regime behavior are so common that it seems worthwhile to study this behavior in more detail. We summarize the findings.

MASTER THEOREM. Let $a \geq 1$ and $b > 1$ be integers and $c \geq 0$ and $d > 0$ real numbers. Let $T(n)$ be defined for integers that are powers of b by

$$T(n) \quad = \quad \begin{cases} aT(\frac{n}{b}) + n^c & \text{if} \quad n > 1 \\ d & \text{if} \quad n = 1. \end{cases}$$

Then we have the following:

- $T(n) = \Theta(n^c)$ if $\log_b a < c$;
- $T(n) = \Theta(n^c \log n)$ if $\log_b a = c$;
- $T(n) = \Theta(n^{\log_b a})$ if $\log_b a > c$.

This behavior can be explained by recalling the formula for a geometric series, $(r^0 + \ldots + r^{n-1})A = \frac{1-r^n}{1-r}A$, and focusing on the magnitude of the constant factor, r. For $0 < r < 1$, the sum is roughly A, the first term, for $r = 1$, the sum is n, the number of terms, and for $r > 1$, the sum is roughly $r^{n-1}A$, the last term.

Let us consider again the recursion tree and, in particular, the total work at its i-th level, starting with $i = 0$ at the root. There are a^i nodes and the work at each node is $(\frac{n}{b^i})^c$. The work at the i-th level is therefore

$$a^i \left(\frac{n}{b^i}\right)^c \quad = \quad n^c \frac{a^i}{b^{ic}}.$$

There are $1 + \log_b n$ levels, and the total work is the sum over the levels. This sum is a geometric series, with factor $r = \frac{a}{b^c}$. It is therefore dominated by the first term if $r < 1$, all terms are the same if $r = 0$, and it is dominated by the last term if $r > 1$. To distinguish between the three cases, we take the logarithm of r, which is negative, zero, positive if $r < 1$, $r = 1$, $r > 1$. It is convenient to take the logarithm to the basis b. This way we get

$$\begin{aligned} \log_b \frac{a}{b^c} \quad &= \quad \log_b a - \log_b b^c \\ &= \quad \log_b a - c. \end{aligned}$$

We have $r < 1$ iff $\log_b a < c$, In which case the dominating term in the series is n^c. We have $r = 1$ iff $\log_b a = c$, in which case the total work is $n^c \log_b n$. We have $r > 1$ iff $\log_b a > c$, in which case the dominating term is $d \cdot a^{\log_b n} = d \cdot n^{\log_b a}$. This explains the three cases in the theorem.

There are extensions of this result that discuss the cases in which n is not a lower of b, we have floors and ceilings in the relation, a and b are not integers, etc. The general behavior of the solution remains the same.

Using induction. Once we know (or feel) what the solution to a recurrence relation is, we can often use induction to verify. Here is a particular relation defined for integers that are powers of 4:

$$T(n) \quad = \quad \begin{cases} T(\frac{n}{2}) + T(\frac{n}{4}) + n & \text{if} \quad n > 1 \\ 1 & \text{if} \quad n = 1. \end{cases}$$

To get a feeling for the solution, we group nodes with equal work together. We get n once, $\frac{n}{2}$ once, $\frac{n}{4}$ twice, $\frac{n}{8}$ three times, $\frac{n}{16}$ five times, etc. These are the Fibonacci numbers, which grow exponentially, with basis equal to the golden ratio, which is roughly 1.6. On the other hand, the work shrinks exponentially, with basis 2. Hence, we have a geometric series with factor roughly 0.8, which is less than one. The dominating term is therefore the first, and we would guess that the solution is some constant times n. We can prove this by induction.

CLAIM. There exists a positive constant c such that $T(n) \leq cn$.

PROOF. For $n = 1$, we have $T(1) = 1$. Hence, the claimed inequality is true provided $c \geq 1$. Using the strong form of Mathematical Induction, we get

$$\begin{aligned} T(n) &= T\left(\frac{n}{2}\right) + T\left(\frac{n}{4}\right) + n \\ &= c\frac{n}{2} + c\frac{n}{4} + n \\ &= \left(\frac{3c}{4} + 1\right) n. \end{aligned}$$

This is at most cn provided $\frac{3c}{4} + 1 \leq c$ or, equivalently, $c \geq 4$. □

The inductive proof not only verified what we thought might be the case, but it also gave us the smallest constant, $c = 4$, for which $T(n) \leq cn$ is true.

Finding the median. Similar recurrence relations arise in practice. A classic example is an algorithm for finding the k-smallest of an unsorted set of n items. We assume the items are all different. A particularly interesting case is the middle item, which is called the *median*. For odd n, this is the k-smallest with $k = \frac{n+1}{2}$. For even n, we set k equal to either the floor or the ceiling of $\frac{n+1}{2}$. The algorithm takes four steps to find the k-smallest item.

STEP 1. Partition the set into groups of size 5 and find the median in each group.

STEP 2. Find the median of the medians.

STEP 3. Split the set into S, the items smaller than the median of the medians, and L, the items larger than the median of the medians.

STEP 4. Let $s = |S|$. If $s < k-1$ then return the $(k-s)$-smallest item in L. If $s = k-1$ then return the median of the medians. if $s > k-1$ then return the k-smallest item in S.

The algorithm is recursive, computing the median of roughly $\frac{n}{5}$ medians in Step 2, and then computing an item either in L or in S. To prove that the algorithm terminates, we need to show that the sets considered recursively get strictly smaller. This is easy as long as n is large but tricky for small n. We ignore these difficulties.

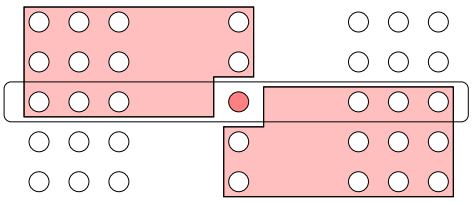


Figure 14: The upper left shaded region consists of items smaller than the median of the medians. Symmetrically, the lower right shaded region consists of items larger than the median of the medians. Both contain about three tenth of all items.

To get a handle on the running time, we need to estimate how much smaller than n the sets S and L are. Consider Figure 14. In one iteration of the algorithm, we eliminate either all items smaller or all items larger than the median of the medians. The number of such items is at least the number in one of the two shaded regions, each containing roughly $\frac{3n}{10}$ items. Hence, the recurrence relation describing the running time of the algorithm is

$$T(n) \quad = \quad \begin{cases} T(\frac{7n}{10}) + T(\frac{n}{5}) + n & \text{if} \quad n > n_0 \\ n_0 & \text{if} \quad n \leq n_0, \end{cases}$$

for some large enough constant n_0. Since $\frac{7}{10} + \frac{1}{5}$ is strictly less than one, we guess that the solution to this recurrence relation is again $\mathrm{O}(n)$. This can be verified by induction.

Fourth Homework Assignment

Write the solution to each problem on a single page. The deadline for handing in solutions is 18 March 2009.

Question 1. ($20 = 10 + 10$ points).

(a) Prove the following claim:

$$1 + 7 + \cdots + (3n^2 - 3n + 1) = n^3.$$

(b) (Problem 4.1-11 in our textbook). Find the error in the following proof that all positive integers n are equal. Let $p(n)$ be the statement that all numbers in an n-element set of positive integers are equal. Then $p(1)$ is true. Let $n \geq 2$ and write N for the set of n first positive integers. Let N' and N'' be the sets of $n-1$ first and $n-1$ last integers in N. By $p(n-1)$, all members of N' are equal, and all members of N'' are equal. Thus, the first $n-1$ elements of N are equal and the last $n-1$ elements of N are equal, and so all elements of N are equal. Therefore, all positive integers are equal.

Question 2. (20 points). Recall the Chinese Remainder Theorem stated for two positive, relatively prime moduli, m and n, in Section 7. Assuming this theorem, prove the following generalization by induction on k.

CLAIM. Let $n_1, n_2, \ldots, n_k$ be positive, pairwise relative prime numbers. Then for every sequence of integers $a_i \in \mathbb{Z}_{n_i}$, $1 \leq i \leq k$, the system of k linear equations,

$$x \bmod n_i \quad = \quad a_i,$$

has a unique solution in $\mathbb{Z}_N$, where $N = \prod_{i=1}^{k} n_i$.

Question 3. ($20 = 10 + 10$ points).

(a) (Problem 4.2-13 in our textbook). Solve the recurrence $T(n) = 2T(n-1) + 3^n$, with $T(0) = 1$.

(b) (Problem 4.2-17 in our textbook). Solve the recurrence $T(n) = rT(n-1)+n$, with $T(0) = 1$. (Assume that $r \neq 1$.)

Question 4. ($20 = 7 + 7 + 6$ points). Consider the following algorithm segment.

```
int FUNCTION(n)
  if n > 0 then
    n = FUNCTION(⌊n/a⌋) + FUNCTION(⌊n/b⌋)
  endif
  return n.
```

We can assume that $a, b > 1$, so the algorithm terminates. In the following questions, let a_n be the number of iterations of the `while` loop.

(a) Find a recurrence relation for a_n.

(b) Find an explicit formula for a_n.

(c) How fast does n grow? (big Θ terms)

Question 5. ($20 = 4+4+4+4+4$ points). (Problem 4.4-1 in our textbook). Use the Master Theorem to solve the following recurrence relations. For each, assume $T(1) = 1$ and n is a power of the appropriate integer.

(a) $T(n) = 8T(\frac{n}{2}) + n$.

(b) $T(n) = 8T(\frac{n}{2}) + n^3$.

(c) $T(n) = 3T(\frac{n}{2}) + n$.

(d) $T(n) = T(\frac{n}{4}) + 1$.

(e) $T(n) = 3T(\frac{n}{3}) + n^2$.

V PROBABILITY

In its simplest form, computing the probability reduces to counting, namely the lucky outcomes and all possible outcomes. The probability is then the ratio of the two, which is a real number between zero and one.

15 Inclusion-Exclusion

Today, we introduce basic concepts in probability theory and we learn about one of its fundamental principles.

Throwing dice. Consider a simple example of a probabilistic experiment: throwing two dice and counting the total number of dots. Each die has six sides with 1 to 6 dots. The result of a throw is thus a number between 2 and 12. There are 36 possible outcomes, 6 for each die, which we draw as the entries of a matrix; see Figure 15.

	1	2	3	4	5	6
1	2	3	4	5	6	7
2	3	4	5	6	7	8
3	4	5	6	7	8	9
4	5	6	7	8	9	10
5	6	7	8	9	10	11
6	7	8	9	10	11	12

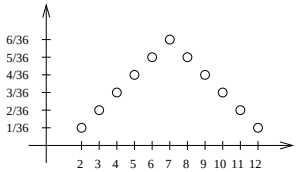


Figure 15: Left: the two dice give the row and the column index of the entry in the matrix. Right: the most likely sum is 7, with probability $\frac{1}{6}$, the length of the diagonal divided by the size of the matrix.

Basic concepts. The set of possible outcomes of an experiment is the *sample space*, denoted as Ω. A possible outcome is an *element*, $x \in \Omega$. A subset of outcomes is an *event*, $A \subseteq \Omega$. The *probability* or *weight* of an element x is $P(x)$, a real number between 0 and 1. For finite sample spaces, the *probability* of an event is $P(A) = \sum_{x \in A} P(x)$.

For example, in the two dice experiment, we set $\Omega = \{2, 3, \ldots, 12\}$. An event could be to throw an even number. The probabilities of the different outcomes are given in Figure 15 and we can compute

$$P(\text{even}) \quad = \quad \frac{1+3+5+5+3+1}{36} \quad = \quad \frac{1}{2}.$$

More formally, we call a function $P : \Omega \to \mathbb{R}$ a *probability distribution* or a *probability measure* if

(i) $P(x) \geq 0$ for every $x \in \Omega$;

(ii) $P(A \dot\cup B) = P(A) + P(B)$ for all disjoint events $A \cap B = \emptyset$;

(iii) $P(\Omega) = 1$.

A common example is the *uniform probability distribution* defined by $P(x) = P(y)$ for all $x, y \in \Omega$. Clearly, if Ω is finite then

$$P(A) \quad = \quad \frac{|A|}{|\Omega|}$$

for every event $A \subseteq \Omega$.

Union of non-disjoint events. Suppose we throw two dice and ask what is the probability that the outcome is even or larger than 7. Write A for the event of having an even number and B for the event that the number exceeds 7. Then $P(A) = \frac{1}{2}$, $P(B) = \frac{15}{36}$, and $P(A \cap B) = \frac{9}{36}$. The question asks for the probability of the union of A and B. We get this by adding the probabilities of A and B and then subtracting the probability of the intersection, because it has been added twice,

$$P(A \cup B) \quad = \quad P(A) + P(B) - P(A \cap B),$$

which gives $\frac{6}{12} + \frac{5}{12} - \frac{3}{12} = \frac{2}{3}$. If we had three events, then we would subtract all pairwise intersections and add back in the triplewise intersection, that is,

$$\begin{aligned} P(A \cup B \cup C) \quad = \quad & P(A) + P(B) + P(C) \\ & -P(A \cap B) - P(A \cap C) \\ & -P(B \cap C) + P(A \cap B \cap C). \end{aligned}$$

Principle of inclusion-exclusion. We can generalize the idea of compensating by subtracting to n events.

PIE THEOREM (FOR PROBABILITY). The probability of the union of n events is

$$P(\bigcup_{i=1}^{n} A_i) \quad = \quad \sum_{k=1}^{n} (-1)^{k+1} \sum P(A_{i_1} \cap \ldots \cap A_{i_k}),$$

where the second sum is over all subsets of k events.

PROOF. Let x be an element in $\bigcup_{i=1}^{n} A_i$ and H the subset of $\{1, 2, \ldots, n\}$ such that $x \in A_i$ iff $i \in H$. The contribution of x to the sum is $P(x)$ for each odd subset of H and $-P(x)$ for each even subset of H. If we include $\emptyset$ as an even subset, then the number of odd and even subsets is the same. We can prove this using the Binomial Theorem:

$$(1-1)^n \quad = \quad \sum_{i=0}^{n} (-1)^i \binom{n}{i}.$$

But in the claimed equation, we do not account for the empty set. Hence, there is a surplus of one odd subset and therefore a net contribution of $P(x)$. This is true for every element. The PIE Theorem for Probability follows. □

Checking hats. Suppose n people get their hats returned in random order. What is the chance that at least one gets the correct hat? Let A_i be the event that person i gets the correct hat. Then

$$P(A_i) \quad = \quad \frac{(n-1)!}{n!} \quad = \quad \frac{1}{n}.$$

Similarly,

$$P(A_{i_1} \cap \ldots \cap A_{i_k}) \quad = \quad \frac{(n-k)!}{n!}.$$

The event that at least one person gets the correct hat is the union of the A_i. Writing $P = P(\bigcup_{i=1}^n A_i)$ for its probability, we have

$$\begin{aligned} P &= \sum_{i=1}^{k} (-1)^{k+1} \sum P(A_{i_1} \cap \ldots \cap A_{i_k}) \\ &= \sum_{i=1}^{k} (-1)^{k+1} \binom{n}{k} \frac{(n-k)!}{n!} \\ &= \sum_{i=1}^{k} (-1)^{k+1} \frac{1}{k!} \\ &= 1 - \frac{1}{2} + \frac{1}{3!} - \ldots \pm \frac{1}{n!}. \end{aligned}$$

Recall from Taylor expansion of real-valued functions that $e^x = 1 + x + x^2/2 + x^3/3! + \ldots$. Hence,

$$P \quad = \quad 1 - e^{-1} = 0.6\ldots$$

Inclusion-exclusion for counting. The principle of inclusion-exclusion generally applies to measuring things. Counting elements in finite sets is an example.

PIE THEOREM (FOR COUNTING). For a collection of n finite sets, we have

$$|\bigcup_{i=1}^{n} A_i| \quad = \quad \sum_{k=1}^{n} (-1)^{k+1} \sum |A_{i_1} \cap \ldots \cap A_{i_k}|,$$

where the second sum is over all subsets of k events.

The only difference to the PIE Theorem for probability is we count one for each element, x, instead of $P(x)$.

Counting surjective functions. Let M and N be finite sets, and $m = |M|$ and $n = |N|$ their cardinalities. Counting the functions of the form $f : M \to N$ is easy. Each $x \in M$ has n choices for its image, the choices are independent, and therefore the number of functions is n^m. How many of these functions are surjective? To answer this question, let $N = \{y_1, y_2, \ldots, y_n\}$ and let A_i be the set of functions in which y_i is not the image of any element in M. Writing A for the set of all functions and S for the set of all surjective functions, we have

$$S \quad = \quad A - \bigcup_{i=1}^{n} A_i.$$

We already know $|A|$. Similarly, $|A_i| = (n-1)^m$. Furthermore, the size of the intersection of k of the A_i is

$$|A_{i_1} \cap \ldots \cap A_{i_k}| \quad = \quad (n-k)^m.$$

We can now use inclusion-exclusion to get the number of functions in the union, namely,

$$|\bigcup_{i=1}^{n} A_i| \quad = \quad \sum_{k=1}^{n} (-1)^{k+1} \binom{n}{k} (n-k)^m.$$

To get the number of surjective functions, we subtract the size of the union from the total number of functions,

$$|S| \quad = \quad \sum_{i=0}^{n} (-1)^k \binom{n}{k} (n-k)^m.$$

For $m < n$, this number should be 0, and for $m = n$, it should be $n!$. Check whether this is indeed the case for small values of m and n.

16 Conditional Probability

If we have partial information, this effectively shrinks the available sample space and changes the probabilities. We begin with an example.

Monty Hall show. The setting is a game show in which a prize is hidden behind one of three curtains. Call the curtains X, Y, and Z. You can win the prize by guessing the right curtain.

STEP 1. You choose a curtain.

This leaves two curtains you did not choose, and at least one of them does not hide the prize. Monty Hall opens this one curtain and this way demonstrates there is no prize hidden there. Then he asks whether you would like to reconsider. Would you?

STEP 2A. You stick to your initial choice.

STEP 2B. You change to the other available curtain.

Perhaps surprisingly, Step 2B is the better strategy. As shown in Figure 16, it doubles your chance to win the prize.

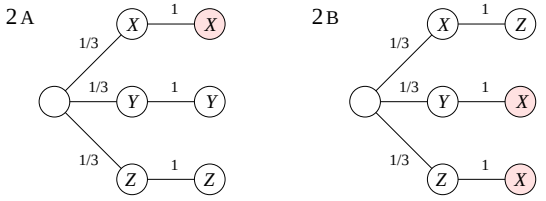


Figure 16: Suppose the prize is behind curtain X. The chance of winning improves from $\frac{1}{3}$ in 2A to $\frac{2}{3}$ in 2B.

Formalization. We are given a sample space, Ω, and consider two events, $A, B \subseteq \Omega$. The *conditional probability* of even A given event B is

$$P(A \mid B) \quad = \quad \frac{P(A \cap B)}{P(B)}.$$

We illustrate this definition in Figure 17. If we know that the outcome of the experiment is in B, the chance that it is also in A is the fraction of B occupied by $A \cap B$. We say A and B are *independent* if knowing B does not change the probability of A, that is,

$$P(A \mid B) \quad = \quad P(A).$$

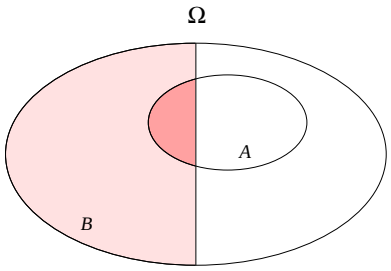


Figure 17: Assuming B, the probability of A is represented by the fraction of the shaded region, B, that is dark shaded, $A \cap B$.

Since $P(A \mid B) = \frac{P(A \cap B)}{P(B)} = P(A)$, we have

$$P(B) \quad = \quad \frac{P(B \cap A)}{P(A)} \quad = \quad P(B \mid A).$$

We thus see that independence is symmetric. However, it fails to be an equivalence relation because it is neither reflexive not transitive. Combining the definition of conditional probability with the condition of independence, we get a formula for the probability of two events occurring at the same time.

PRODUCT PRINCIPLE FOR INDEPENDENT PROB. If A and B are independent then $P(A \cap B) = P(A) \cdot P(B)$.

Trial processes. In many situations, a probabilistic experiment is repeated, possibly many times. We call this a *trial process*. It is *independent* if the i-th trial is not influenced by the outcomes of the preceding $i-1$ trials, that is,

$$P(A_i \mid A_1 \cap \ldots \cap A_{i-1}) \quad = \quad P(A_i),$$

for each i.

An example is picking a coin from an bag that contains one nickel, two dimes, and two quarters. We have an independent trial process if we always return the coin before the next draw. The choice we get a quarter is therefore $\frac{2}{5}$ each time. The chance to pick the quarter three times in a row is therefore $(\frac{2}{5})^3 = \frac{8}{125} = 0.064$. More generally, we have the

INDEPENDENT TRIAL THEOREM. In an independent trial process, the probability of a sequence of outcomes, $a_1, a_2, \ldots, a_n$, is $P(a_1) \cdot P(a_2) \cdot \ldots \cdot P(a_n)$.

Trial processes that are not independent are generally more complicated and we need more elaborate tools to

compute the probabilities. A useful such tool is the tree diagram as shown in Figure 18 for the coin picking experiment in which we do not replace the picked coins.

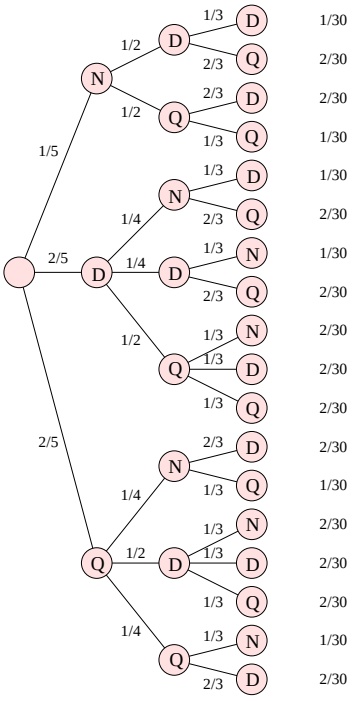


Figure 18: What is the probability of picking the nickel in three trials?

Medical test example. Probabilities can be counterintuitive, even in situations in which they are important. Consider a medical test for a disease, D. The test mostly gives the right answer, but not always. Say its false-negative rate is 1% and its false-positive rate is 2%, that is,

$$\begin{aligned} P(\text{y} \mid D) &= 0.99; \\ P(\text{n} \mid D) &= 0.01; \\ P(\text{y} \mid \neg D) &= 0.02; \\ P(\text{n} \mid \neg D) &= 0.98. \end{aligned}$$

Assume that the chance you have disease D is only one in a thousand, that is, $P(D) = 0.001$. Now you take the test and the outcome is positive. What is the chance that you have the disease? In other words, what is $P(D \mid \text{y})$? As illustrated in Figure 19,

$$P(D \mid \text{y}) = \frac{P(D \cap \text{y})}{P(\text{y})} = \frac{0.00099}{0.02097} = 0.047\ldots.$$

This is clearly a case in which you want to get a second opinion before starting a treatment.

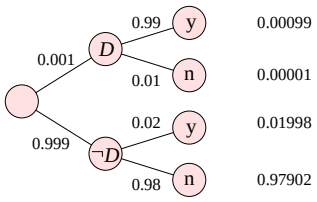


Figure 19: Tree diagram showing the conditional probabilities in the medical test question.

Summary. Today, we learned about conditional probabilities and what it means for two events to be independent. The Product Principle can be used to compute the probability of the intersection of two events, if they are independent. We also learned about trial processes and tree diagrams to analyze them.

17 Random Variables

A *random variable* is a real-value function on the sample space, $X : \Omega \to \mathbb{R}$. An example is the total number of dots at rolling two dice, or the number of heads in a sequence of ten coin flips.

Bernoulli trial process. Recall that an independent trial process is a sequence of identical experiments in which the outcome of each experiment is independent of the preceding outcomes. A particular example is the *Bernoulli trial process* in which the probability of success is the same at each trial:

$$\begin{aligned} P(\text{success}) &= p; \\ P(\text{failure}) &= 1-p. \end{aligned}$$

If we do a sequence of n trials, we may define X equal to the number of successes. Hence, Ω is the space of possible outcomes for a sequence of n trials or, equivalently, the set of binary strings of length n. What is the probability of getting exactly k successes? By the Independent Trial Theorem, the probability of having a sequence of k successes followed by $n-k$ failures is $p^k(1-p)^{n-k}$. Now we just have to multiply with the number of binary sequences that contain k successes.

BINOMIAL PROBABILITY LEMMA. The probability of having exactly k successes in a sequence of n trials is $P(X=k) = \binom{n}{k} p^k (1-p)^{n-k}$.

As a sanity check, we make sure that the probabilities add up to one. Using the Binomial Theorem, get

$$\sum_{k=0}^{n} P(X=k) = \sum_{k=0}^{n} \binom{n}{k} p^k (1-p)^{n-k},$$

which is equal to $(p+(1-p))^n = 1$. Because of this connection, the probabilities in the Bernoulli trial process are called the *binomial probabilities*.

Expectation. The function that assigns to each $x_i \in \mathbb{R}$ the probability that $X = x_i$ is the *distribution function* of X, denoted as $f : \mathbb{R} \to [0,1]$; see Figure 20. More formally, $f(x_i) = P(A)$, where $A = X^{-1}(x_i)$. The *expected value* of the random variable is $E(X) = \sum_i x_i P(X = x_i)$.

As an example, consider the Bernoulli trial process in which X counts the successes in a sequence of n trials,

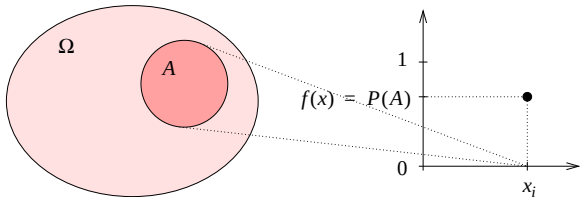


Figure 20: The distribution function of a random variable is constructed by mapping a real number, x_i, to the probability of the event that the random variable takes on the value x_i.

that is, $P(X=k) = \binom{n}{k} p^k (1-p)^{n-k}$. The corresponding distribution function maps k to the probability of having k successes, that is, $f(k) = \binom{n}{k} p^k (1-p)^{n-k}$. We get the expected number of successes by summing over all k.

$$\begin{aligned} E(X) &= \sum_{k=0}^{n} k f(k) \\ &= \sum_{k=0}^{n} k \binom{n}{k} p^k (1-p)^{n-k} \\ &= np \sum_{k=1}^{n} \binom{n-1}{k-1} p^{k-1} (1-p)^{n-k} \\ &= np \sum_{k=0}^{n-1} \binom{n-1}{k} p^k (1-p)^{n-k-1}. \end{aligned}$$

The sum in the last line is equal to $(p+(1-p))^{n-1} = 1$. Hence, the expected number of successes is $X = np$.

Linearity of expectation. Note that the expected value of X can also be obtained by summing over all possible outcomes, that is,

$$E(X) = \sum_{s \in \Omega} X(s) P(s).$$

This leads to an easier way of computing the expected value. To this end, we exploit the following important property of expectations.

LINEARITY OF EXPECTATION. Let $X, Y : \Omega \to \mathbb{R}$ be two random variables. Then

(i) $E(X+Y) = E(X) + E(Y)$;

(ii) $E(cX) = cE(X)$, for every real number c.

The proof should be obvious. Is it? We use the property to recompute the expected number of successes in

a Bernoulli trial process. For i from 1 to n, let X_i be the expected number of successes in the i-th trial. Since there is only one i-th trial, this is the same as the probability of having a success, that is, $E(X_i) = p$. Furthermore, $X = X_1 + X_2 + \ldots + X_n$. Repeated application of property (i) of the Linearity of Expectation gives $E(X) = \sum_{i=1}^{n} E(X_i) = np$, same as before.

Indication. The Linearity of Expectation does not depend on the independence of the trials; it is also true if X and Y are dependent. We illustrate this property by going back to our hat checking experiment. First, we introduce a definition. Given an event, the corresponding *indicator random variable* is 1 if the event happens and 0 otherwise. Thus, $E(X) = P(X = 1)$.

In the hat checking experiment, we return n hats in a random order. Let X be the number of correctly returned hats. We proved that the probability of returning at least one hat correctly is $P(X \geq 1) = 1 - e^{-1} = 0.6\ldots$ To compute the expectation from the definition, we would have to determine the probability of returning exactly k hats corrects, for each $0 \leq k \leq n$. Alternatively, we can compute the expectation by decomposing the random variable, $X = X_1 + X_2 + \ldots + X_n$, where X_i is the expected value that the i-th hat is returned correctly. Now, X_i is an indicator variable with $E(X_i) = \frac{1}{n}$. Note that the X_i are not independent. For example, if the first $n-1$ hats are returned correctly then so is the n-th hat. In spite of the dependence, we have

$$E(X) = \sum_{i=1}^{n} E(X_i) = 1.$$

In words, the expected number of correctly returned hats is one.

Example: computing the minimum. Consider the following algorithm for computing the minimum among n items stored in a linear array.

```
min = A[1];
for i = 2 to n do
  if min > A[i] then min = A[i] endif
endif.
```

Suppose the items are distinct and the array stores them in a random sequence. By this we mean that each permutation of the n items is equally likely. Let X be the number of assignments to min. We have $X = X_1 + X_2 + \ldots + X_n$, where X_i is the expected number of assignments in the i-th step. We get $X_i = 1$ iff the i-th item, $A[i]$, is smaller than all preceding items. The chance for this to happen is one in i. Hence,

$$\begin{aligned} E(X) &= \sum_{i=1}^{n} E(X_i) \\ &= \sum_{i=1}^{n} \frac{1}{i}. \end{aligned}$$

The result of this sum is referred to as the *n-th harmonic number*, $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}$. We can use $\int_{x=1}^{n} = \ln n$ to show that the n-th harmonic number is approximately the natural logarithm of n. More precisely, $\ln(n+1) \leq H_n \leq 1 + \ln n$.

Waiting for success. Suppose we have again a Bernoulli trial process, but this time we end it the first time we hit a success. Defining X equal to the index of the first success, we are interested in the expected value, $E(X)$. We have $P(X = i) = (1-p)^{i-1}p$ for each i. As a sanity check, we make sure that the probabilities add up to one. Indeed,

$$\begin{aligned} \sum_{i=1}^{\infty} P(X = i) &= \sum_{i=1}^{\infty} (1-p)^{i-1} p \\ &= p \cdot \frac{1}{1-(1-p)}. \end{aligned}$$

Using the Linearity of Expectation, we get a similar sum for the expected number of trials. First, we note that $\sum_{j=0}^{\infty} jx^j = \frac{x}{(1-x)^2}$. There are many ways to derive this equation, for example, by index transformation. Hence,

$$\begin{aligned} E(X) &= \sum_{i=0}^{\infty} iP(X = i) \\ &= \frac{p}{1-p} \sum_{i=0}^{\infty} i(1-p)^i \\ &= \frac{p}{1-p} \cdot \frac{1-p}{(1-(1-p))^2}, \end{aligned}$$

which is equal to $\frac{1}{p}$.

Summary. Today, we have learned about random variable and their expected values. Very importantly, the expectation of a sum of random variables is equal to the sum of the expectations. We used this to analyze the Bernoulli trial process.

18 Probability in Hashing

A popular method for storing a collection of items to support fast look-up is hashing them into a table. Trouble starts when we attempt to store more than one item in the same slot. The efficiency of all hashing algorithms depends on how often this happens.

Birthday paradox. We begin with an instructive question about birthdays. Consider a group of n people. Each person claims one particular day of the year as her birthday. For simplicity, we assume that nobody claims February 29 and we talk about years consisting of $k = 365$ days only. Assume also that each day is equally likely for each person. In other words,

$$P(\text{person } i \text{ is born on day } j) \quad = \quad \frac{1}{k},$$

for all i and all j. Collecting the birthdays of the n people, we get a multiset of n days during the year. We are interested in the event, A, that at least two people have the same birthday. Its probability is one minus the probability that the n birthdays are distinct, that is,

$$\begin{aligned} P(A) &= 1 - P(\bar{A}) \\ &= 1 - \frac{k}{k} \cdot \frac{k-1}{k} \cdot \ldots \cdot \frac{k-n+1}{k} \\ &= 1 - \frac{k!}{(k-n)!k^n}. \end{aligned}$$

The probability of A surpasses one half when n exceeds 21, which is perhaps surprisingly early. See Figure 21 for a display how the probability grows with increasing n.

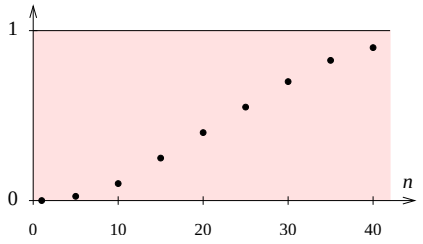


Figure 21: The probability that at least two people in a group of n share the same birthday.

Hashing. The basic mechanism in hashing is the same as in the assignment of birthdays. We have n items and map each to one of k slots. We assume the n choices of slots are independent. A *collision* is the event that an item is mapped to a slot that already stores an item. A possible resolution of a collision adds the item at the end of a linked list that belongs to the slot, but there are others. We are interested in the following quantities:

1. the expected number of items mapping to same slot;
2. the expected number of empty slots;
3. the expected number of collisions;
4. the expected number of items needed to fill all k slots.

Different hashing algorithms use different mechanisms for resolving collisions. The above quantities have a lot to say about the relative merits of these algorithms.

Items per slot. Since all slots are the same and none is more preferred than any other, we might as well determine the expected number of items that are mapped to slot 1. Consider the corresponding indicator random variable,

$$X_i \quad = \quad \begin{cases} 1 & \text{if item } i \text{ is mapped to slot 1;} \\ 0 & \text{otherwise.} \end{cases}$$

The number of items mapped to slot 1 is therefore $X = X_1 + X_2 + \ldots + X_n$. The expected value of X_i is $\frac{1}{k}$, for each i. Hence, the expected number of items mapped to slot 1 is

$$E(X) \quad = \quad \sum_{i=1}^{n} E(X_i) \quad = \quad \frac{n}{k}.$$

But this is obvious in any case. As mentioned earlier, the expected number of items is the same for every slot. Writing Y_j for the number of items mapped to slot j, we have $Y = \sum_{j=1}^{k} Y_j = n$. Similarly,

$$E(Y) \quad = \quad \sum_{j=1}^{k} E(Y_j) \quad = \quad n.$$

Since the expectations are the same for all slots, we therefore have $E(Y_j) = \frac{n}{k}$, for each j.

Empty slots. The probability that slot j remains empty after mapping all n items is $(1 - \frac{1}{k})^n$. Defining

$$X_j \quad = \quad \begin{cases} 1 & \text{if slot } j \text{ remains empty;} \\ 0 & \text{otherwise,} \end{cases}$$

we thus get $E(X_j) = (1 - \frac{1}{k})n$. The number of empty slots is $X = X_1 + X_2 + \ldots + X_k$. Hence, the expected

number of empty slots is

$$E(X) \;=\; \sum_{j=1}^{k} E(X_j) \;=\; k\left(1-\frac{1}{k}\right)^n.$$

For $k=n$, we have $\lim_{n\to\infty}(1-\frac{1}{n})^n = e^{-1} = 0.367\ldots$ In this case, we can expect about a third of the slots to remain empty.

Collisions. The number of collisions can be determined from the number of empty slots. Writing X for the number of empty slots, as before, we have $k-X$ items hashed without collision and therefore a total of $n-k+X$ collisions. Writing Z for the number of collisions, we thus get

$$\begin{aligned} E(Z) &= n-k+E(X) \\ &= n-k+k\left(1-\frac{1}{k}\right)^n. \end{aligned}$$

For $k=n$, we get $\lim_{n\to\infty} n(1-\frac{1}{n})^n = \frac{n}{e}$. In words, about a third of the items cause a collision.

Filling all slots. How many items do we need to map to the k slots until they store at least one item each? For obvious reasons, this question is sometimes referred to as the coupons collector problem. The crucial idea here is to define X_j equal to the number of items it takes to go from $j-1$ to j filled slots. Filling the j-th slot is an infinite Bernoulli process with success probability equal to $p = \frac{k-j+1}{k}$. Last lecture, we learned that the expected number of trials until the first success is $\frac{1}{p}$. Hence, $E(X_j) = \frac{k}{k-j+1}$. The number of items needed to fill all slots is $X = X_1 + X_2 + \ldots + X_k$. The expected number is therefore

$$\begin{aligned} E(X) &= \sum_{j=1}^{k} E(X_j) \\ &= \sum_{j=1}^{k} \frac{k}{k-j+1} \\ &= k\sum_{j=1}^{k} \frac{1}{j} \\ &= kH_k. \end{aligned}$$

As mentioned during last lecture, this is approximately k times the natural logarithm of k. More precisely, we have $k\ln(k+1) \le kH_k \le k(1+\ln k)$.

19 Probability Distributions

Although individual events based on probability are unpredictable, we can predict patterns when we repeat the experiment many times. Today, we will look at the pattern that emerges from independent random variables, such as flipping a coin.

Coin flipping. Suppose we have a fair coin, that is, the probability of getting head is precisely one half and the same is true for getting tail. Let X count the times we get head. If we flip the coin n times, the probability that we get k heads is

$$P(X = k) \quad = \quad \binom{n}{k} / 2^n.$$

Figure 22 visualizes this distribution in the form of a histogram for $n = 10$. Recall that the *distribution function* maps every possible outcome to its probability, $f(k) = P(X = k)$. This makes sense when we have a discrete domain. For a continuous domain, we consider the *cumulative distribution function* that gives the probability of the outcome to be within a particular range, that is, $\int_{x=a}^{b} f(x)\,\mathrm{d}x = P(a \leq X \leq b)$.

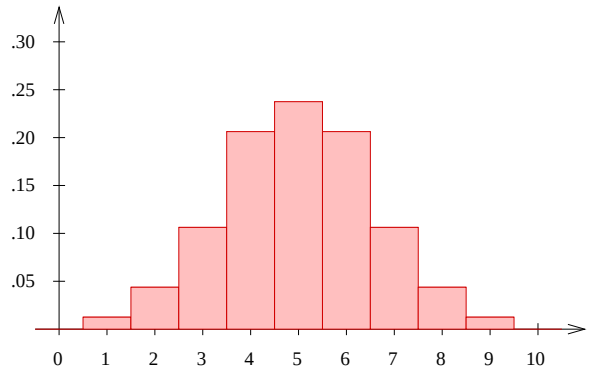


Figure 22: The histogram that the shows the probability of getting $0, 1, \ldots, 10$ heads when flipping a coin ten times.

Variance. Now that we have an idea of what a distribution function looks like, we wish to find succinct ways of describing it. First, we note that $\mu = E(X)$ is the expected value of our random variable. It is also referred to as the *mean* or the *average* of the distribution. In the example above, where X is the number of heads in ten coin flips, we have $\mu = 5$. However, we would not be surprised if we had four or six heads but we might be surprised if we had zero or ten heads when we flip a coin ten times. To express how surprised we should be we measure the spread of the distribution. Let us first determine how close we expect a random variable to be to its expectation, $E(X - E(X))$. By linearity of expectation, we have

$$E(X - \mu) \quad = \quad E(X) - E(\mu) \quad = \quad \mu - \mu \quad = \quad 0.$$

Hence, this measurement is not a good description of the distribution. Instead, we use the expectation of the square of the difference to the mean. Specifically, the *variance* of a random variable X, denoted as $V(X)$, is the expectation $E\left((X - \mu)^2\right)$. The *standard deviation* is the square root of the variance, that is, $\sigma(X) = V(X)^{1/2}$. If X_4 is the number of heads we see in four coin flips, then $\mu = 2$ and

$$V(X_4) \quad = \quad \frac{1}{16}\left[(-2)^2 + 4 \cdot (-1)^2 + 4 \cdot 1^2 + 2^2\right],$$

which is equal to 1. For comparison, let X_1 be the number of heads that we see in one coin flip. Then $\mu = \frac{1}{2}$ and

$$V(X_1) \quad = \quad \frac{1}{2}\left[(0 - \frac{1}{2})^2 + (1 - \frac{1}{2})^2\right],$$

which is equal to one quarter. Here, we notice that the variance of four flips is the sum of the variances for four individual flips. However, this property does not hold in general.

Variance for independent random variables. Let X and Y be independent random variables. Then, the property that we observed above is true.

ADDITIVITY OF VARIANCE. If X and Y are independent random variables then $V(X + Y) = V(X) + V(Y)$.

We first prove the following more technical result.

LEMMA. If X and Y are independent random variables then $E(XY) = E(X)E(Y)$.

PROOF. By definition of expectation, $E(X)E(Y)$ is the product of $\sum_i x_i P(X = x_i)$ and $\sum_j y_j P(Y = y_j)$. Pushing the summations to the right, we get

$$\begin{aligned} E(X)E(Y) \quad &= \quad \sum_i \sum_j x_i y_j P(X = x_i) P(Y = y_j) \\ &= \quad \sum_{i,j} z_{ij} P(X = x_i) P(Y = y_j), \end{aligned}$$

where $z_{ij} = x_i y_j$. Finally, we use the independence of the random variables X and Y to see that $P(X = x_i)P(Y = y_j) = P(XY = z_{ij})$. With this, we conclude that $E(X)E(Y) = E(XY)$. □

Now, we are ready to prove the Additivity of Variance, that is, $V(X+Y) = V(X) + V(Y)$ whenever X and Y are independent.

PROOF. By definition of variance, we have

$$V(X+Y) = E\left((X+Y-E(X+Y))^2\right).$$

The right hand side is the expectation of $(X-\mu_X)^2 + 2(X-\mu_X)(Y-\mu_Y) + (Y-\mu_Y)$, where μ_X and μ_Y are the expected values of the two random variables. With this, we get

$$\begin{aligned} V(X+Y) &= E\left((X-\mu_X)^2\right) + E\left((Y-\mu_Y)^2\right) \\ &= V(X) + V(Y), \end{aligned}$$

as claimed. □

Normal distribution. If we continue to increase the number of coin flips, then the distribution function approaches the *normal distribution*,

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}.$$

This is the limit of the distribution as the number of coin flips approaches infinity. For a large number of trials, the

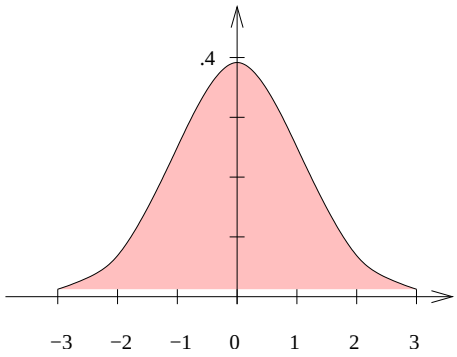


Figure 23: The normal distribution with mean $\mu = 0$ and standard deviation $\sigma = 1$. The probability that the random variable is between $-\sigma$ and σ is 0.68, between -2σ and 2σ is 0.955, and between -3σ and 3σ is 0.997.

normal distribution can be used to approximate the probability of the sum being between a and b standard deviations from the expected value.

STANDARD LIMIT THEOREM. The probability of the number of heads being between $a\sigma$ and $b\sigma$ from the mean goes to

$$\frac{1}{\sqrt{2\pi}} \int_{x=a}^{b} e^{-\frac{x^2}{2}} \, \mathrm{d}x$$

as the number of flips goes to infinity.

For example, if we have 100 coin flips, then $\mu = 50$, $V(X) = 25$, and $\sigma = 5$. It follows that the probability of having between 45 and 55 heads is about 0.68.

Summary. We used a histogram to visualize the probability that we will have k heads in n flips of a coin. We also used the mean, μ, the standard deviation, σ, and the variance, $V(X)$, to describe the distribution of outcomes. As n approaches infinity, we see that this distribution approaches the normal distribution.

Fifth Homework Assignment

Write the solution to each problem on a single page. The deadline for handing in solutions is 8 April 2009.

Question 1. (20 points). Use the Principle of Inclusion-Exclusion to count the surjective functions $f : M \to N$, where where both sets are finite with $m = |M|$ and $n = |N|$.

Question 2. ($20 = 6 + 7 + 7$ points). (Problems 5.3-1 to 3 in our textbook). Suppose you have a fair coin, one in which a flip gives head with probability one half and tail with probability one half. You do three flips with this coin.

(a) What is the probability that two flips in are row are heads, given that there is an even number of heads?

(b) Is the event that two flips in a row are heads independent of the event that there is an even number of heads?

(c) Is the event of getting at most one tail independent of the event that not all flips are identical?

Question 3. (20 points). (Problem 5.4-16 in our textbook). Suppose you have two nickels, two dimes, and two quarters in a bag. You draw three coins from the bag, without replacement. What is the expected amount of money you get?

Question 4. ($20 = 6 + 7 + 7$ points). (Problem 5.5-8 in our textbook). Suppose you hash n items into k locations.

(a) What is the probability that all n items has to different locations?

(b) What is the probability that the i-th item gives the first collision?

(c) What is the expected number of items you hash until the first collision?

Question 5. ($20 = 7 + 7 + 6$ points). In the programming language of your choice, write the following two functions:

(a) GETMEAN

(a) GETVARIANCE

These methods should take an array of values as input (the experimental results for each trial) and return a floating point number. Then, flip a coin 20 times (or simulate this on the computer) and use these methods to compute the mean and the variance of your trials. Are the results what you would have expected?

Question 6. ($20 = 10 + 10$ points). (Problems 5.7-8 and 14 in our textbook).

(a) Show that if X and Y are independent, and b and c are constant, then $X - b$ and $Y - c$ are independent.

(b) Given a random variable X, how does the variance of cX relate to that of X?

VI GRAPHS

A graph is a set of vertices with pairs connected by edges. The information in a particular graph is contained in the choice of vertices that are connected by edges. This simple combinatorial structure is surprisingly versatile and conveniently models situations in which the relationship between parts is important.

20 Trees

Graphs can be used to model and solve many problems. Trees are special graphs. Today, we look at various properties of graphs as well as of trees.

Party problem. Suppose we choose six people at random from a party. We call the people A through F. Then, one of the following must be true:

I. three of the people mutually know each other; or

II. three of the people mutually do not know each other.

We reformulate this claim using a graph representing the situation. We draw six vertices, one for each person, and we draw an edge between two vertices if the two people know each other. We call this a *simple graph*. The *complement graph* consists of the same six vertices but contains an edge between two vertices iff the graph does not have an edge between them. Property I says the graph contains a triangle and Property II says the complement graph contains a triangle. In Figure 24, we see two graphs on six vertices. On the left, the graph contains a triangle and on the right, the complement graph contains a triangle. We can now reformulate the above claim.

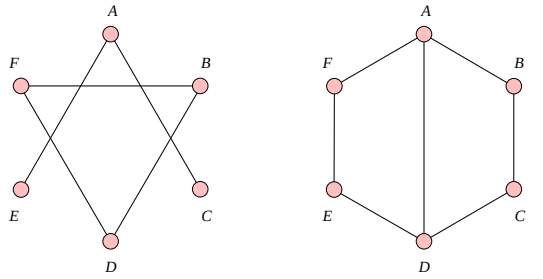


Figure 24: The two cases we consider in the party problem.

PARTY THEOREM. If a simple graph on six vertices does not have a triangle then the complement graph on the same six vertices has a triangle.

PROOF. We distinguish the case in which A knows two or fewer people from the case in which A knows three or more people. For the first case, we assume that A possibly knows C and E but nobody else. If A knows both and C and E know each other, then we have a triangle. Otherwise, consider B, D, F. If they do not all mutually know each other, then without loss of generality, we can say that B does not know D. Thus, we have a triangle in the complement graph since A, B, and D mutually do not know each other. For the second case, assume that A knows B, D, F, and possibly other people. If any two of the three know each other then we have a triangle in the graph. Otherwise, we have a triangle in the complement graph since B, D, and F mutually do not know each other. □

Simple graphs. In the above problem, we used graphs to help us find the solution. A *(simple) graph*, $G = (V, E)$, is a finite set of vertices, V, together with a set of edges, E, where an edge is an unordered pair of vertices. Two vertices connected by an edge are *adjacent* and they are *neighbors* of each other. Furthermore, we say the edge is *incident* to the vertices it connects. Sometimes we refer to the vertices as nodes and the edges as arcs. For example, Figure 25 shows the *complete graph* of five vertices, which we denote as K_5. This graph is complete since we cannot add any more edges. A *subgraph* of a graph $G = (V, E)$ is

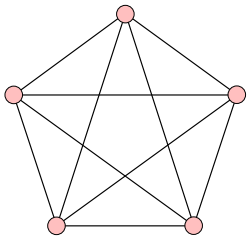


Figure 25: The complete graph of five vertices. It has $\binom{5}{2} = 10$ edges.

a graph $H = (W, F)$ for which $W \subseteq V$ and $F \subseteq E$. For example, a *clique* in G is a subgraph that is a complete graph if considered by itself. With this terminology, the Party Theorem says that a graph on six vertices contains a clique of three vertices or its complement graph contains such a clique.

Connectivity. Suppose we have a graph where the nodes are cities and an edge $\{a, b\}$ exists if there is a train that goes between these two cities. Where can you go if you start at a city x? We need some definitions to study this question. A *walk* is an alternating sequence of vertices and edges such that

1. the sequence starts and ends with a vertex;
2. each edge connects the vertex that precedes the edge with the vertex that succeeds the edge.

Furthermore, a *path* is a walk such that no vertex appears twice. If there exists a walk from a to b, then we know that there also exists a path from a to b. Indeed, if a vertex x appears twice, we can shorten the walk by removing all edges and vertices between the two copies as well as one copy of the vertex x. If the walk is finite, we get a path after finitely many operations as described.

CLAIM. Having a connecting path is an equivalence relation on the vertices of a graph.

PROOF. Let a, b, c be three vertices of a graph G. The relation is reflexive because (a) is a path from a to a. The relation is symmetric because reversing a path from a to b gives a path from b to a. Finally, the relation is transitive because concatenating a path from a to b with a path from b to c gives a path from a to c. □

A graph is *connected* if there is a path between every pair of its vertices. A *connected component* of a not necessarily connected graph is a maximal connected subgraph. Equivalently, a connected component is an equivalence class of vertices together with the edges between them.

Cycles and trees. A *closed walk* is a walk that starts and ends at the same vertex. A *cycle* is a closed walk in which no vertices are repeated. Alternatively, we can say that a cycle is a path in which the start and the end vertices are the same. A *tree* is a connected graph that does not have any cycles.

PROPERTIES OF TREES. If $T = (V, E)$ is a tree with n vertices and m edges, then we have the following properties:

1. there is exactly one path between every pair of vertices;
2. removing an edge disconnects the tree;
3. the number of edges is $m = n - 1$;
4. there exists at least one vertex that has precisely one neighboring vertex.

Spanning trees. Sometimes the whole graph gives us more information than we need or can process. In such a case, we may reduce the graph while preserving the property of interest. For example, if we are interested in connectivity, we may remove as many edges as we can while preserving the connectivity of the graph. This leads to the concept of a *spanning tree*, that is, a subgraph that contains all vertices and is itself a tree.

SPANNING TREE THEOREM. Every finite connected graph contains a spanning tree.

PROOF. If there is a cycle in the graph, remove one edge of the cycle. Repeat until no cycles remain. □

There are a number of different algorithms for constructing a spanning tree. We may, for example, begin with a vertex $u_0 \in V$ and grow a tree by adding an edge and a vertex in each round. This is called Prim's Algorithm.

$W = \{u_0\}$; $F = \emptyset$; $X = V - \{u_0\}$;
`while` $\exists$ edge $\{w, x\}$ with $w \in W$ and $x \in X$ `do`
  move x from X to W; add $\{w, x\}$ to F
`endwhile`;
`if` $V = W$ `then` (W, F) is spanning tree
  `else` (V, E) is not connected
`endif`.

At the end of this algorithm, we have determined if G is connected. If it is connected, we have found a spanning tree. Otherwise, (W, F) is a spanning tree of the connected component that contains u_0.

Rooted trees. In many situations, it is convenient to have the edges of a tree directed, from one endpoint to the other. If we have an edge from a to b, we call a a *parent* of b and b a *child* of a. A particularly important such structure is obtained if the edges are directed such that each vertex has at most one parent. In a tree, the number of edges is one less than the number of vertices. This implies that each vertex has exactly one parent, except for one, the *root*, which has no parent. Holding the root and letting gravity work on the rest of the graph, we get a picture like in Figure 26 in which the root is drawn at the top. The

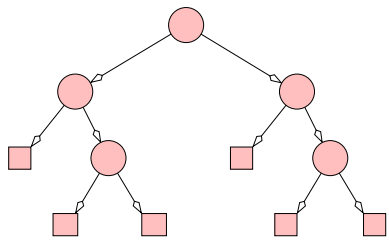


Figure 26: The top vertex is the root and the square vertices are the leaves of the rooted tree.

directions of the edges are now determined, namely from top to bottom, leading away from the root. Each maximal directed path starts at the root and ends at a *leaf*, that is,

a vertex without children. Rooted trees are often used to model or to support a search process. We start at the root and choose an outgoing edge to direct the search to one of the available subtrees. We then recurse, treating the new vertex like the root. Repeating this step, we eventually arrive at a leaf of the rooted tree. Similarly, we can give a recursive definition of a rooted tree. We phrase this definition of the case of a *binary tree*, that is, a rooted tree in which every vertex has at most two children. We thus talk about a left child and a right child.

- an empty graph is a binary tree;
- a vertex (the root) with a binary tree as left subtree and a binary tree as right subtree is a binary tree.

While uncommon in Mathematics, recursive definitions are suggestive of algorithms and thus common in Computer Science.

Summary. Today, we looked at graphs, subgraphs, trees, and rooted trees. We used Prim's algorithm to find a spanning tree (if one exists).

21 Tours

In this section, we study different ways to traverse a graph. We begin with tours that traverse every edge exactly once and end with tours that visit every vertex exactly once.

Bridges of Königsberg. The Pregel River goes through the city of Königsberg, separating it into two large islands and two pieces of mainland. There are seven bridges connecting the islands with the mainland, as sketched in Figure 27. Is it possible to find a closed walk that traverses

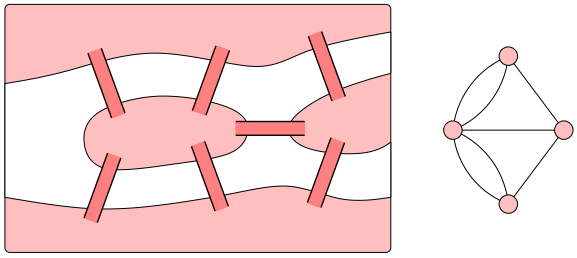


Figure 27: Left: schematic picture of the bridges connecting the islands with the mainland in Königsberg. Right: representation by a graph with four vertices and seven edges.

each bridge exactly once? We can formalize this question by drawing a graph with four vertices, one for each island and each piece of the mainland. We have an edge for each bridge, as in Figure 27 on the right. The graph has *multi-edges* and is therefore not simple. More generally, we may also allow *loops* that are edges starting and ending at the same vertex. A *Eulerian tour* of such a graph is a closed walk that contains each edge exactly once.

Eulerian graphs. A graph is *Eulerian* if it permits a Eulerian tour. To decide whether or not a graph is Eulerian, it suffices to look at the local neighborhood of each vertex. The *degree* of a vertex is the number of incident edges. Here we count a loop twice because it touches a vertex at both ends.

EULERIAN TOUR THEOREM. A graph is Eulerian iff it is connected and every vertex has even degree.

PROOF. If a graph is Eulerian then it is connected and each vertex has even degree just because we enter a vertex the same number of times we leave it. The other direction is more difficult to prove. We do it constructively. Given a vertex $u_0 \in V$, we construct a maximal walk, W_0, that leaves each vertex at a yet unused edge. Starting at u_0, the walk continues until we have no more edge to leave the last vertex. Since each vertex has even degree, this last vertex can only be u_0. The walk W_0 is thus necessarily closed. If it is not a Eulerian tour then there are still some unused edges left. Consider a connected component of the graph consisting of these unused edges and the incident vertices. It is connected and every vertex has even degree. Let u_1 be a vertex of this component that also lies on W_0. Construct a closed walk, W_1, starting from u_1. Now concatenate W_0 and W_1 to form a longer closed walk. Repeating this step a finite number of times gives a Eulerian tour. □

All four vertices of the graph modeling the seven bridges in Königsberg have odd degree. It follows there is no closed walk that traverses each bridge exactly once.

Hamiltonian graphs. Consider the *pentagon dodecahedron*, the Platonic solid bounded by twelve faces, each a regular pentagon. Drawing the corners as vertices and the sides of the pentagons as edges, we get a graph as in Figure 28. Recall that a cycle in a graph is a closed walk

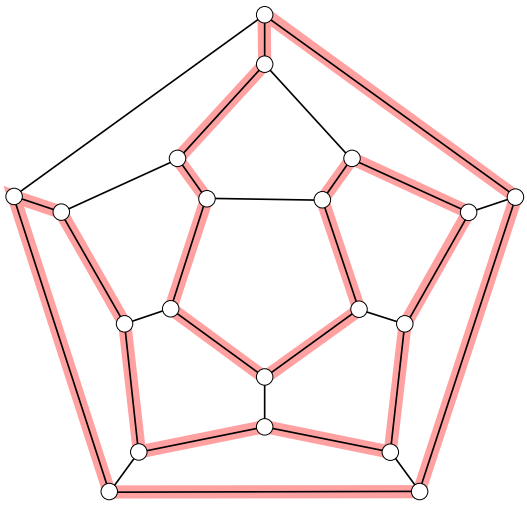


Figure 28: A drawing of a pentagon dodecahedron in which the lengths of the edges are not in scale.

in which no vertex is repeated. A *Hamiltonian cycle* is a closed walk that visits every vertex exactly once. As indicated by the shading of some edges in Figure 28, the graph of the pentagon dodecahedron has a Hamiltonian cycle. A graph is *Hamiltonian* if it permits a Hamiltonian cycle. Deciding whether or not a graph is Hamiltonian turns out to be much more difficult than deciding whether or not it is Eulerian.

A sufficient condition. The more edges we have, the more likely it is to find a Hamiltonian cycle. It turns out that beyond some number of edges incident to each vertex, there is always a Hamiltonian cycle.

DIRAC'S THEOREM. If G is a simple graph with $n \geq 3$ vertices and each vertex has degree at least $\frac{n}{2}$ then G is Hamiltonian.

PROOF. Assume G has a maximal set of edges without being Hamiltonian. Letting x and y be two vertices not adjacent in G, we thus have a path from x to y that passes through all vertices of the graph. We index the vertices along this path, with $u_1 = x$ and $u_n = y$, as in Figure 29. Now suppose x is adjacent to a vertex u_{i+1}. If y is

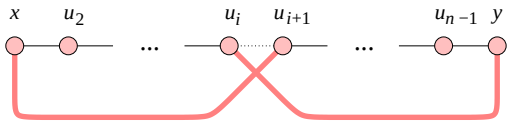


Figure 29: If x is adjacent to u_{i+1} and y is adjacent to u_i then we get a Hamiltonian cycle by adding these two edges to the path and removing the edge connecting u_i to u_{i+1}.

adjacent to u_i then we have a Hamiltonian cycle as shown in Figure 29. Thus, for every neighbor u_{i+1} of x, we have a non-neighbor u_i of y. But x has at least $\frac{n}{2}$ neighbors which implies that y has at least $\frac{n}{2}$ non-neighbors. The degree of y is therefore at most $(n-1) - \frac{n}{2} = \frac{n}{2} - 1$. This contradicts the assumption and thus implies the claim. □

The proof of Dirac's Theorem uses a common technique, namely assuming an extreme counterexample and deriving a contradiction from this assumption.

Summary. We have learned about Eulerian graphs which have closed walks traversing each edge exactly once. Such graphs are easily recognized, simply by checking the degree of all the vertices. We have also learned about Hamiltonian graphs which have closed walks visiting each vertex exactly once. Such graphs are difficult to recognize. More specifically, there is no known algorithm that can decide whether a graph of n vertices is Hamiltonian in time that is at most polynomial in n.

22 Matching

Most of us are familiar with the difficult problem of finding a good match. We use graphs to study the problem from a global perspective.

Marriage problem. Suppose there are n boys and n girls and we have a like-dislike relation between them. Representing this data in a square matrix, as in Figure 30 on the left, we write an 'x' whenever the corresponding boy and girl like each other. Alternatively, we may represent the data in form of a graph in which we draw an edge for each 'x' in the matrix; see Figure 30 on the right. This graph, $G = (V, E)$, is *bipartite*, that is, we can partition the vertex set as $V = X \dot\cup Y$ such that each edge connects a vertex in X with a vertex in Y. The sets X and Y are the *parts* of the graph.

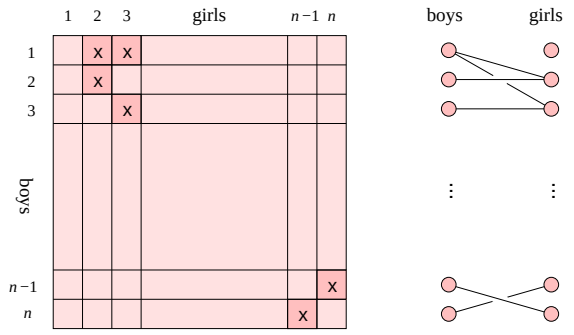


Figure 30: The matrix on the left and the bipartite graph on the right both represent the same data.

The goal is to marry off the boys and girls based on the relation. We formalize this using the bipartite graph representation. A *matching* is a set $M \subseteq E$ of vertex-disjoint edges. The matching is *maximal* if no matching properly contains M. The matching is *maximum* if no matching has more edges than M. Note that every maximum matching is maximal but not the other way round. Maximal matchings are easy to find, eg. by greedily adding one edge at a time. To construct a maximum matching efficiently, we need to know more about the structure of matchings.

Augmenting paths. Let $G = (V, E)$ be a bipartite graph with parts X and Y and $M \subseteq E$ a matching. An *alternating path* alternates between edges in M and edges in $E - M$. An *augmenting path* is an alternating path that begins and ends at unmatched vertices, that is, at vertices that are not incident to edges in M. If we have an augmenting path, we can switch its edges to increase the size of the matching, as in Figure 31.

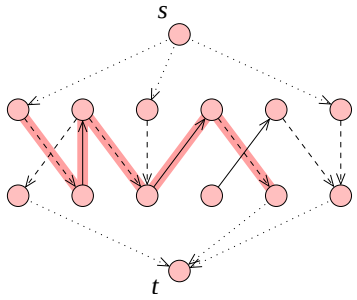


Figure 31: The solid edges form a matching. The shaded edges form an augmenting path. Trading its dashed for its solid edges, we increase the size of the matching by one edge. If we add s and t and direct the edges, the augmenting path becomes a directed path connecting s to t.

BERGE'S THEOREM. The matching M is maximum iff there is no augmenting path.

PROOF. Clearly, if there is an augmenting path then M is not maximum. Equivalently, if M is maximum then there is no augmenting path. Proving the other direction is more difficult. Suppose M is not maximum. Then there exists a matching N with $|N| > |M|$. We consider the symmetric difference obtained by removing the duplicate edges from their union,

$$M \oplus N \quad = \quad (M \cup N) - (M \cap N).$$

Since both sets are matchings, the edges of M are vertex-disjoint and so are the edges of N. It follows that each connected component of the graph $(V, M \oplus N)$ is either an alternating path or an alternating cycle. Every alternating cycle has the same number of edges from M and from N. Since N has more edges than M, it also contributes more edges to the symmetric difference. Hence, at least one component has more edges from N than from M. This is an augmenting path. □

Constructing a maximum matching. Berge's Theorem suggests we construct a maximum matching iteratively. Starting with the empty matching, $M = \emptyset$, we find an augmenting path and increase the size of the matching in each iteration until no further increase is possible. The number of iterations is less than the number of vertices. To explain

how we find an augmenting path, we add vertices s and t to the graph, connecting s to all unmatched vertices in X and t to all unmatched vertices in Y. Furthermore, we direct all edges: from s to X, from X to Y if the edge is in $E-M$, from Y to X if the edge is in M, and from Y to t; see Figure 31. An augmenting path starts and ends with an edge in $E-M$. Prepending an edge from s and appending an edge to t, we get a directed path from s to t in the directed graph. Such a path can be found with breadth-first search, which works by storing active vertices in a queue and marking vertices that have already been visited. Initially, s is the only marked vertex and the only vertex in the queue. In each iteration, we remove the last vertex, x, from the end of the queue, mark all unmarked successors of x, and add these at the front to the queue. We halt the algorithm when t is added to the queue. If this never happens then there is no augmenting path and the matching is maximum. Otherwise, we extract a path from s to t by tracing it from the other direction, starting at t, adding one marked vertex at a time.

The breadth-first search algorithm takes constant time per edge. The number of edges is less than n^2, where $n = |V|$. It follows that an augmenting path can be found in time $O(n^2)$. The overall algorithms takes time $O(n^3)$ to construct a maximum matching.

Vertex covers. Running the algorithm to completion, we get a maximum matching, $M \subseteq E$. Let Y_0 contain all vertices in Y reachable from s and X_0 all vertices in X from which t is reachable; see Figure 32. No edge in M

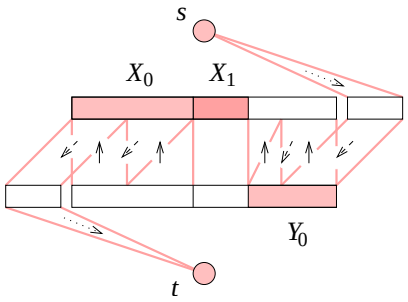


Figure 32: Schematic picture of the vertex set D consisting of the shaded portions of X and of Y. The vertices are ordered so that all edges in M are vertical.

connects a vertex in X_0 with a vertex in Y_0, else we would have an augmenting path. Furthermore, $|X_0 \cup Y_0| \leq |M|$ because each vertex in the union is incident to an edge in the matching. Letting X_1 contain the endpoints of the yet untouched edges in M, we set $D = X_0 \cup Y_0 \cup X_1$ and note that $|D| = |M|$. Furthermore, we observe that D *covers* all edges in E, that is, each edge has at least one endpoint in D.

We generalize this concept. Given a graph $G = (V, E)$, a *vertex cover* is a set $C \subseteq V$ such that each edge in E has at least one endpoint in C. It is *minimal* if it does not properly contain another vertex cover and *minimum* if there is no vertex cover with fewer vertices. Finding a minimal vertex cover is easy, eg. by greedily removing one vertex at a time, but finding a minimum vertex cover is a difficult computational problem for which no polynomial-time algorithm is known. However, if G is bipartite, we can use the maximum matching algorithm to construct a minimum vertex cover.

KÖNIG'S THEOREM. If $G = (V, E)$ is bipartite then the size of a minimum vertex cover is equal to the size of a maximum matching.

PROOF. Let X and Y be the parts of the graph, $C \subseteq V = X \dot\cup Y$ a minimum vertex cover, and $M \subseteq E$ a maximum matching. Then $|M| \leq |C|$ because C covers M. Since M is maximum, there is no augmenting path. It follows that the set $D \subseteq V$ (as defined above) covers all edges. Since C is minimum, we have $|C| \leq |D| = |M|$, which implies the claim. □

Neighborhood sizes. If the two parts of the bipartite graph have the same size it is sometimes possible to match every last vertex. We call a matching *perfect* if $|M| = |X| = |Y|$. There is an interesting relationship between the existence of a perfect matching and the number of neighbors a set of vertices has. Let $S \subseteq X$ and define its *neighborhood* as the set $N(S) \subseteq Y$ consisting of all vertices adjacent to at least one vertex in S.

HALL'S THEOREM. In a bipartite graph $G = (V, E)$ with equally large parts X and Y, there is a perfect matching iff $|N(S)| \geq |S|$ for every $S \subseteq X$.

PROOF. If all vertices of X can be matched then $|N(S)| \geq |S|$ simply because $N(S)$ contains all matched vertices in Y, and possibly more. The other direction is more difficult to prove. We show that $|N(S)| \geq |S|$ for all $S \subseteq X$ implies that X is a minimum vertex cover. By König's Theorem, there is a matching of the same size, and this matching necessarily connects to all vertices in X.

Let now $C \subseteq X \dot\cup Y$ be a minimum vertex cover and consider $S = X - C$. By definition of vertex cover, all

neighbors of vertices in S are in $Y \cap C$. Hence, $|S| \leq |N(S)| \leq |Y \cap C|$. We therefore have

$$\begin{aligned} |C| &= |C \cap X| + |C \cap Y| \\ &\geq |C \cap X| + |S| \\ &= |C \cap X| + |X - C| \end{aligned}$$

which is equal to $|X|$. But X clearly covers all edges, which implies $|C| = |X|$. Hence, X is a minimum vertex cover, which implies the claim. □

Summary. Today, we have defined the marriage problem as constructing a maximum matching in a bipartite graph. We have seen that such a matching can be constructed in time cubic in the number of vertices. We have also seen connections between maximum matchings, minimum vertex covers, and sizes of neighborhoods.

23 Planar Graphs

Although we commonly draw a graph in the plane, using tiny circles for the vertices and curves for the edges, a graph is a perfectly abstract concept. We now talk about constraints on graphs necessary to be able to draw a graph in the plane without crossings between the curves. This question forms a bridge between the abstract and the geometric study of graphs.

Drawings and embeddings. Let $G = (V, E)$ be a simple, undirected graph and let $\mathbb{R}^2$ denote the two-dimensional real plane. A *drawing* maps every vertex $u \in V$ to a point $\varepsilon(u)$ in $\mathbb{R}^2$, and it maps every edge $uv \in E$ to a curve with endpoints $\varepsilon(u)$ and $\varepsilon(v)$; see Figure 33. The drawing is an *embedding* if

1. vertices are mapped to distinct points;
2. edge are mapped to curves without self-intersections;
3. a curve does not pass through a point, unless the corresponding edge and vertex are incident, in which case the point is an endpoint of the curve;
4. two curves are disjoint, unless the corresponding edges are incident to a common vertex, in which case the curves share a common endpoint.

Not every graph can be drawn without crossings between the curves. The graph G is *planar* if it has an embedding in the plane.

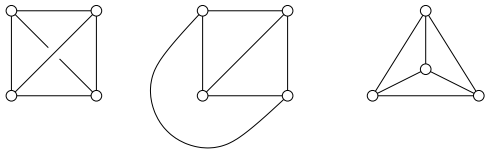


Figure 33: Three drawings of K_4. From left to right a drawing that is not an embedding, an embedding with one curved edge, and a straight-line embedding.

Euler's Formula. Think of the plane as an infinite piece of paper which you cut along the curves with a pair of scissors. Each piece of the paper that remains connected after the cutting is called a *face* of the embedding. We write $n = |V|$, $m = |E|$, and ℓ for the number of faces. Euler's Formula is a linear relation between the three numbers.

EULER'S FORMULA. For an embedding of a connected graph we have $n - m + \ell = 2$.

PROOF. Choose a spanning tree (V, T) of (V, E). It has n vertices, $|T| = n - 1$ edges, and one face. We have $n - (n - 1) + 1 = 2$, which proves the formula if G is a tree. Otherwise, draw the remaining edges, one at a time. Each edge decomposes one face into two. The number of vertices does not change, m increases by one, and ℓ increases by one. Since the graph satisfies the claimed linear relation before drawing the edge it satisfies the relation also after drawing the edge. □

We get bounds on the number of edges and faces, in terms of the number of vertices, by considering *maximally connected* graphs for which adding any other edge would violate planarity. Every face of a maximally connected planar graph with three or more vertices is necessarily a triangle, for if there is a face with more than three edges we can add an edge without crossing any other edge. Let $n \geq 3$ be the number of vertices, as before. Since every face has three edges and every edge belong to two triangles, we have $3\ell = 2m$. We use this relation to rewrite Euler's Formula: $n - m + \frac{2m}{3} = 2$ and $n - \frac{3\ell}{2} + \ell = 2$ and therefore $m = 3n - 6$ and $\ell = 2n - 4$. Every planar graph can be completed to a maximally connected planar graph, which implies that it has at most these numbers of edges and faces.

Note that the sum of vertex degrees is twice the number of edges, and therefore $\sum_u \deg(u) \leq 6n - 12$. It follows that every planar graph has a vertex of degree less than six. We will see uses of this observation in coloring planar graphs and in proving that they have straight-line embeddings.

Non-planarity. We can use the consequences of Euler's Formula to prove that the complete graph of five vertices and the complete bipartite graph of three plus three vertices are not planar. Consider first K_5, which is drawn in Figure 34, left. It has $n = 5$ vertices and $m = 10$ edges,

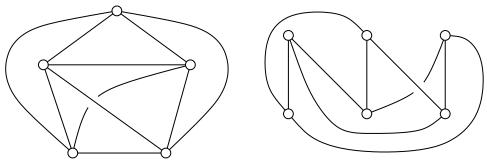


Figure 34: K_5 on the left and $K_{3,3}$ on the right.

contradicting the upper bound of at most $3n-6 = 9$ edges for maximally connected planar graphs. Consider second $K_{3,3}$, which is drawn in Figure 34, right. It has $n = 6$ vertices and $m = 9$ edges. Each cycle has even length,

which implies that each face has four or more edges. We get $4\ell \leq 2m$ and $m \leq 2n - 4 = 8$ after plugging the inequality into Euler's Formula, again a contradiction.

In a sense, K_5 and $K_{3,3}$ are the quintessential non-planar graphs. Two graphs are *homeomorphic* if one can be obtained from the other by a sequence of operations, each deleting a degree-2 vertex and merging its two edges into one or doing the inverse.

KURATOWSKI'S THEOREM. A graph G is planar iff no subgraph of G is homeomorphic to K_5 or to $K_{3,3}$.

The proof of this result is a bit lengthy and omitted. We now turn to two applications of the structural properties of planar graphs we have learned.

Vertex coloring. A *vertex k-coloring* is a map $\chi : V \to \{1, 2, \ldots, k\}$ such that $\chi(u) \neq \chi(v)$ whenever u and v are adjacent. We call $\chi(u)$ the *color* of the vertex u. For planar graphs, the concept is motivated by coloring countries in a geographic map. We model the problem by replacing each country by a vertex and by drawing an edge between the vertices of neighboring countries. A famous result is that every planar graph has a 4-coloring, but proving this fills the pages of a thick book. Instead, we give a constructive argument for the weaker result that every planar graph has a 5-coloring. If the graph has five or fewer vertices then we color them directly. Else we perform the following four steps:

`Step 1.` Remove a vertex $u \in V$ with degree $k = \deg(u) \leq 5$, together with the k incident edges.

`Step 2.` If $k = 5$ then find two neighbors v and w of the removed vertex u that are not adjacent and merge them into a single vertex.

`Step 3.` Recursively construct a 5-coloring of the smaller graph.

`Step 4.` Add u back into the graph and assign a color that is different from the colors of its neighbors.

Why do we know that vertices v and w in Step 2 exist? To see that five colors suffice, we just need to observe that the at most five neighbors of u use up at most four colors. The idea of removing a small-degree vertex, recursing for the remainder, and adding the vertex back is generally useful. We show that it can also be used to construct embeddings with straight edges.

Convexity and star-convexity. We call a region S in the plane *convex* if for all points $x, y \in S$ the line segment with endpoints x and y is contained in S. Figure 35 shows examples of regions of either kind. We call S *star-convex*

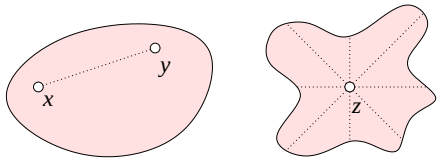


Figure 35: A convex region on the left and a non-convex star-convex region on the right.

if there is a point $z \in S$ such that for every point $x \in S$ the line segment connecting x with z is contained in S. The set of such points z is the *kernel* of S.

It is not too difficult to show that every pentagon is star-convex: decompose the pentagon using two diagonals and choose z close to the common endpoint of these diagonals, as shown in Figure 36. Note however that not every hexagon is star-convex.

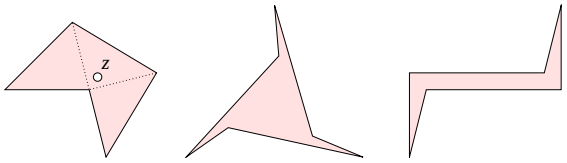


Figure 36: A (necessarily) star-convex pentagon and two non-star-convex hexagons.

Straight-line embedding. A *straight-line embedding* maps every (abstract) edge to the straight line segment connecting the images of its two vertices. We prove that every planar graph has a straight-line embedding using the fact that it has a vertex of degree at most five. To simplify the construction, we assume that the planar graph G is maximally connected and we fix the 'outer' triangle abc. Furthermore, we observe that if G has at least four vertices then it has a vertex of degree at most 5 that is different from a, b and c. Indeed, the combined degree of a, b, c is at least 7. The combined degree of the other $n-3$ vertices is therefore at most $6n - 19$, which implies the average is still less than 6, as required.

`Step 1.` Remove a vertex $u \in V - \{a, b, c\}$ with degree $k = \deg(u) \leq 5$, together with the k incident

edges. Add $k-3$ edges to make the graph maximally connected again.

`Step 2.` Recursively construct a straight-line embedding of the smaller graph.

`Step 3.` Remove the added $k-3$ edges and map u to a point $\varepsilon(u)$ inside the kernel of the k-gon. Connect $\varepsilon(u)$ with line segments to the vertices of the k-gon.

Figure 37 illustrates the recursive construction. It would be fairly straightforward to turn the construction into a recursive algorithm, but the numerical quality of the embeddings it gives is not great.

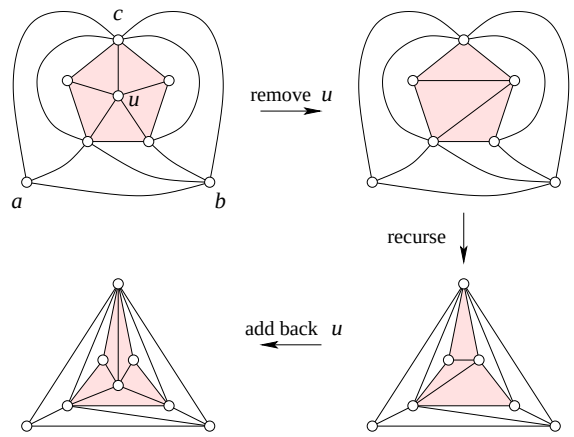


Figure 37: We fix the outer triangle abc, remove the degree-5 vertex u, recursively construct a straight-line embedding of the rest, and finally add the vertex back.

Sixth Homework Assignment

Write the solution to each problem on a single page. The deadline for handing in solutions is 22 April 2009.

Question 1. ($20 = 5 + 5 + 5 + 5$ points). Choose ten of your friends, and make a graph where the edges represent two friends being Facebook friends. (Do not include yourself in the graph). Order your friends alphabetically, and label the vertices $v_1, v_2, \ldots, v_{10}$ respectively. This will be most interesting if all of your friends know each other. Now, answer the following questions about the graph that you drew.

(a) What is the size of the largest clique?

(b) Find the shortest and longest paths from v_1 to v_{10}.

(c) Which vertex has the highest degree?

(d) Use Prim's algorithm to find the minimum spanning tree, and draw that tree.

Question 2. (20 points). (Problem 6.1-14 in our textbook). Are there graphs with n vertices and $n - 1$ edges and no cycles that are not trees? Give a proof to justify your answer.

Question 3. (20 points). Call a simple graph with $n \geq 3$ vertices an *Ore graph* if every pair of non-adjacent vertices has a combined degree of at least n. Is it true that every Ore graph is Hamiltonian? Justify your answer.

Question 4. ($20 = 10 + 10$ points). (Problems 6.4-12 and 13 in our textbook). Prove or give a counterexample:

(a) Every tree is a bipartite graph.

(b) A bipartite graph has no odd cycles.

Question 5. ($20 = 5 + 15$ points). Suppose you have n pennies which you arrange flat on a table, without overlap.

(a) How would you arrange the pennies to maximize the number of pennies that touch each other?

(b) Prove that the number of touching pairs cannot exceed $3n$.